Chapter 9
Model-Based Pose Estimation

Gerard Pons-Moll and Bodo Rosenhahn

Abstract Model-based pose estimation algorithms aim at recovering human motion from one or more camera views and a 3D model representation of the human body. The model pose is usually parameterized with a kinematic chain and thereby the pose is represented by a vector of joint angles. The majority of algorithms are based on minimizing an error function that measures how well the 3D model fits the image. This category of algorithms usually has two main stages, namely defining the model and fitting the model to image observations. In the first section, the reader is introduced to the different kinematic parametrization of human motion. In the second section, the most commonly used representations of the human shape are described. The third section is dedicated to the description of different error functions proposed in the literature and to common optimization techniques used for human pose estimation. Specifically, local optimization and particle-based optimization and filtering are discussed and compared. The chapter concludes with a discussion of the state-of-the-art in model-based pose estimation, current limitations and future directions.

9.1 Kinematic Parametrization

In this chapter our main concern will be on estimating the human pose from images. Human motion is mostly articulated, i.e., it can be accurately modeled by a set of connected rigid segments. A *segment* is a set of points that move rigidly together. To determine the pose, we must first find an appropriate parametrization of the human motion. For the task of estimating human motion a *good* parametrization must have the following attributes.

G. Pons-Moll (✉) · B. Rosenhahn
Leibniz University, Hanover, Germany
e-mail: pons@tnt.uni-hannover.de

B. Rosenhahn
e-mail: rosenhahn@tnt.uni-hannover.de

Attributes of a good parametrization for human motion:

- Pose configurations are represented with the minimum number of parameters.
- Human motion constraints, such as articulated motion, are naturally described.
- Singularities can be avoided during optimization.
- Easy computation of derivatives of segment positions and orientations w.r.t. the parameters.
- Simple rules for concatenating motions.

A commonly used parametrization that meets most of the above requirements is a kinematic chain, which encodes the motion of a body segment as the motion of the previous segment in the chain and an angular motion about a body joint. For example, the motion of the lower arm is parametrized as the motion of the upper arm and a rotation about the elbow. The motion of a body segment relative to the previous one is parametrized by a rotation. Parameterizing rotations can be tricky since it is a non-Euclidean group, which means that if we travel any integer number of loops around an axis in space we will end up in the same point. We now briefly review the different parametrization of rotations that have been used for human tracking.

9.1.1 Rotation Matrices

A rotation matrix $\mathbf{R}_{3\times3}$ is an element of $SO(3)$. Elements of $\mathbf{R} \in SO(3)$ are the group of 3×3 orthonormal matrices with $\det(\mathbf{R}) = 1$ that represent rotations [34]. A rotation matrix encodes the orientation of a frame B that we call *body frame* relative to a second one S that we call *spatial frame*. Given a point $\mathbf{p}$ with body coordinates, $\mathbf{p}_b = (\lambda_x, \lambda_y, \lambda_z)^T$, we might write the point $\mathbf{p}$ in spatial coordinates as

$$\mathbf{p}_s = \lambda_x \mathbf{x}_s^B + \lambda_y \mathbf{y}_s^B + \lambda_z \mathbf{z}_s^B, \tag{9.1}$$

where $\mathbf{x}_s^B$, $\mathbf{y}_s^B$, $\mathbf{z}_s^B$ are the principal axis of the body frame B written in spatial coordinates. We may also write the relationship between the spatial and body frame coordinates in matrix form as $\mathbf{p}_s = \mathbf{R}_{sb}\mathbf{p}_b$. From this it follows that the rotation matrix is given by

$$\mathbf{R}_{sb} = \begin{bmatrix} \mathbf{x}_s^B & \mathbf{y}_s^B & \mathbf{z}_s^B \end{bmatrix}. \tag{9.2}$$

Now consider a frame B whose origin is translated w.r.t. frame S by $\mathbf{t}_s$ (the translation vector written in spatial coordinates). In this case, the coordinates of frames S and B are related by a rotation and a translation, $\mathbf{p}_s = \mathbf{R}_{sb}\mathbf{p}_b + \mathbf{t}_s$. Hence, a pair ($\mathbf{R} \in SO(3)$, $\mathbf{t} \in \mathbb{R}^3$) determines the configuration of a frame B relative to another S and is the product space of $\mathbb{R}^3$ with $SO(3)$ denoted as $SE(3) =$

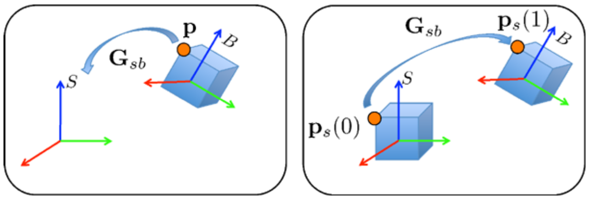

Fig. 9.1 *Left*: rigid body motion seen as a coordinate transformation. *Right*: rigid body motion seen as a relative motion in time

$\mathbb{R}^3 \times SO(3)$. Elements of $SE(3)$ are $\mathbf{g} = \{\mathbf{R}, \mathbf{t}\}$. Equivalently, writing the point in homogeneous coordinates $\bar{\mathbf{p}}_b = \left[\begin{smallmatrix}\mathbf{p}_b \\ 1\end{smallmatrix}\right]$ allows us to use the more compact notation

$$\bar{\mathbf{p}}_s = \mathbf{G}_{sb}\bar{\mathbf{p}}_b, \quad \text{where } \mathbf{G}_{sb} = \begin{bmatrix} \mathbf{R}_{sb[3\times3]} & \mathbf{t}_{s[3\times1]} \\ \mathbf{0}_{[1\times3]} & 1 \end{bmatrix}. \tag{9.3}$$

The rigid body motion is then completely represented by the matrix $\mathbf{G}_{sb}$ which is the homogeneous representation of $\mathbf{g}_{sb}$. The reader unfamiliar with rotation matrices might be surprised because the definitions given here for rotation and rigid motion do not represent motion of points in a fixed frame but rather transformations between coordinate systems. This does not correspond to our informal understanding of rotations. Consequently, *do rotations and rigid body motion represent coordinate transformations or motion?* The answer is: both. To see this, consider a point $\mathbf{p}$ in a rigid body, see Fig. 9.1, and imagine that the body and spatial frames coincide at $t = 0$ see Fig. 9.1 right, consequently $\mathbf{p}_s(0) = \mathbf{p}_b$. At this time we apply the rigid body motion to the point such that the point now moves to a new position $\mathbf{p}_s(1)$. We can write it as

$$\bar{\mathbf{p}}_s(1) = \mathbf{G}_{sb}\bar{\mathbf{p}}_s(0), \tag{9.4}$$

where the coordinates of $\mathbf{p}_s(1)$ and $\mathbf{p}_s(0)$ are both relative to the spatial frame. This new interpretation of rigid motion will be very useful when we talk about human motion in the next section. Both interpretations of rigid motion are correct and depending on the context one is preferable over the other, (e.g., to think about world-to-camera mapping it is better to interpret it as a coordinate transformation but when we think of human motion it is most of the times more intuitive to think of rigid motion as the relative motion between temporal instants). Rotations can be combined by simple matrix multiplication. However, representing rotations with rotation matrices is suboptimal for optimization problems. This is because from the nine numbers composing the matrix, six additional constraints must be imposed during optimization in order to ensure that the matrix is orthonormal. Therefore, representing angular motions with rotation matrices is problematic for motion tracking because we need more parameters than strictly needed.

9.1.2 Euler Angles

One method for describing the orientation of a frame B relative to another frame S is as follows: start with frame B coincident with frame S, rotate B α degrees about the x-axis of frame S, then rotate β degrees about the y-axis of frame S and finally rotate γ degrees about the z-axis (of frame S again). This corresponds to the x, y, z Euler angles defined in frame S. There are several conventions on the order in which these rotations are carried out; for example, it is also possible to perform the rotation in the order z, y, z. Therefore, when we talk about Euler angles the order of the rotations must be specified. It is very important to note with respect to which frame the rotations are defined, they can be defined on the fixed reference frame S or alternatively on the moving frame B. Therefore, a rotation matrix can always be written as the composition of three rotations around the x, y, z axes (9.5). Note that had we chosen the rotations to be defined in the moving frame B, the order would be inverted.

$$\mathbf{R} = \begin{bmatrix} \cos(\gamma) & -\sin(\gamma) & 0 \\ \sin(\gamma) & \cos(\gamma) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\beta) & 0 & -\sin(\beta)) \\ 0 & 1 & 0 \\ \sin(\beta) & 0 & \cos(\beta) \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) \\ 0 & \sin(\alpha) & \cos(\alpha) \end{bmatrix}. \tag{9.5}$$

In this manner, a rotation is completely defined by a triplet of Euler angles (α, β, γ). The derivatives of a rotation with respect to the Euler angles are easy to compute. Additionally, differential equation integration in parameter space is straightforward, for example to update one of the three angles: $\alpha_t = \alpha_{t-1} + \dot{\alpha}$. Unfortunately, Euler angles have a well known problem: when two of the rotation axis align one of the rotations is lost. This well known singularity of Euler parametrization is called gimbal lock.

9.1.3 Quaternions

Quaternions generalize complex numbers and can be used to represent 3D rotations the same way as complex numbers can be used to represent planar rotations. Formally, a quaternion is a vector quantity of the form $\mathbf{q} = q_w + q_x \cdot \mathbf{i} + q_y \cdot \mathbf{j} + q_z \cdot \mathbf{k}$ with $\mathbf{i}^2 = \mathbf{j}^2 = \mathbf{k}^2 = \mathbf{i} \cdot \mathbf{j} \cdot \mathbf{k} = -1$. Unit length quaternions form a set called S^3 which can be used to carry out rotations. They can also be interpreted as a scalar q_w plus a 3-vector $(q_w, \mathbf{v})$. One nice property about quaternions is that rotations can be carried out in parameter space via quaternion product. Given two quaternions $\mathbf{q}_1 = (q_{w,1}, \mathbf{v}_1)$ and $\mathbf{q}_2 = (q_{w,2}, \mathbf{v}_2)$ the quaternion product denoted by $(\circ)$ is defined as

$$\mathbf{q}_1 \circ \mathbf{q}_2 = (q_{w,1} q_{w,2} - \mathbf{v}_1 \cdot \mathbf{v}_2,\ q_{w,1}\mathbf{v}_2 + q_{w,2}\mathbf{v}_1 + \mathbf{v}_1 \times \mathbf{v}_2). \tag{9.6}$$

If we want to rotate a vector $\mathbf{a}$ we can simply use the quaternion product. Thereby, a rotation by an angle θ about an axis ω is represented by the quaternion:

$$\mathbf{q} = [q_w, q_x, q_y, q_z]^T = \left(\cos\left(\frac{\theta}{2}\right), \omega \sin\left(\frac{\theta}{2}\right)\right) \tag{9.7}$$

and the vector $\mathbf{a}$ is rotated with

$$\mathbf{a}' = Rotate(\mathbf{a}) = \mathbf{q} \circ \tilde{\mathbf{a}} \circ \bar{\mathbf{q}}, \tag{9.8}$$

where $\circ$ denotes quaternion product, $\tilde{\mathbf{a}} = [0, \mathbf{a}]$ is a zero scalar component appended with the original vector $\mathbf{a}$ and $\bar{\mathbf{q}} = (q_w, -\mathbf{v})$ is the complex conjugate of $\mathbf{q}$. Additionally, there exist simple formulas for computing the rotation matrix from a quaternion and vice versa. Furthermore, the four partial derivatives $\frac{\partial \mathbf{R}}{\partial q_w}, \frac{\partial \mathbf{R}}{\partial q_x}, \frac{\partial \mathbf{R}}{\partial q_y}, \frac{\partial \mathbf{R}}{\partial q_z}$ exist and are linearly independent in S^3 which means there are no singularities. Probably this last property is the most interesting but this comes at the expense of using 4 numbers instead of just 3. This means that during optimization we must impose a quadratic constraint so that the quaternion keeps unit length. Integrating ODEs can also be problematic since the quaternion velocity $\dot{\mathbf{q}}$ generally lies in the tangent space of S^3 and any movement in the tangent plane will push the quaternion *off* S^3. Nonetheless, there exist solutions to these limitations [25, 41]. Since unit quaternions directly represent the space of rotations and are free of singularities they provide an efficient representation of rotations. Particularly, quaternions have proven to be very useful for the interpolation of key-frame poses because they respect $SO(3)$ geometry.

9.1.4 Axis–Angle

To model human joint motion it is often needed to specify the axis of rotation of the joint. For example we might want to specify the motion of the knee joint as a rotation about an axis perpendicular to the leg and parallel to the hips. Therefore, for our purpose the axis–angle representation is optimal because rotations are described as an angle θ and an axis in space $\omega \in \mathbb{R}^3$ where θ determines the amount of rotation about ω. Unlike quaternions the axis–angle, requires only three parameters $\theta\omega$ to describe a rotation. It does not suffer from gimbal lock and their singularities occur in a region of parameter space that can be easily avoided. Since it will be our parametrization of choice to model human joint motion we will give a brief introduction to the formulation of twists and exponential maps. For a more detailed description we refer the reader to [34].

9.1.4.1 The Exponential Formula

Every rotation $\mathbf{R}$ can be written in exponential form in terms of the axis of rotation $\omega \in \mathbb{R}^3$ and the angle of rotation θ as

$$\mathbf{R} = \exp(\theta\widehat{\omega}), \tag{9.9}$$

where $\widehat{\omega} \in so(3)$ is the skew symmetric matrix constructed from ω. The elements of $so(3)$ are skew symmetric matrices i.e., matrices that verify $\{\mathbf{A} \in \mathbb{R}^{3\times 3} | \mathbf{A} = -\mathbf{A}^T\}$.

Given the vector $\theta(\omega_1, \omega_2, \omega_3)$ the skew symmetric matrix is constructed with the wedge operator $\wedge$ as follows:

$$\theta\widehat{\omega} = \theta \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}. \tag{9.10}$$

By definition, the multiplication of the matrix $\widehat{\omega}$ with a point $\mathbf{p}$ is equivalent to the cross product of the vector ω with the point.

To derive the exponential formula in (9.9) consider a 3D point $\mathbf{p}$ rotating about an axis ω intersecting the origin at a unit constant angular velocity. Recall from elementary physics that the tangential velocity of the point may be written as

$$\dot{\mathbf{p}}(t) = \omega \times \mathbf{p}(t) = \widehat{\omega}\mathbf{p}(t) \tag{9.11}$$

which is a differential equation that we can integrate to obtain

$$\mathbf{p}(t) = \exp(\widehat{\omega}t)\mathbf{p}(0). \tag{9.12}$$

It follows that if we rotate θ units of time the net rotation is given by

$$\mathbf{R}(\theta, \omega) = \exp(\theta\widehat{\omega}). \tag{9.13}$$

The exponential map of a matrix $\mathbf{A} \in \mathbb{R}^{n\times m}$ is analogous to the exponential used for real numbers $a \in \mathbb{R}$. In particular the Taylor expansion of the exponential has the same form:

$$\exp(\theta\widehat{\omega}) = e^{(\theta\widehat{\omega})} = I + \theta\widehat{\omega} + \frac{\theta^2}{2!}\widehat{\omega}^2 + \frac{\theta^3}{3!}\widehat{\omega}^3 + \cdots. \tag{9.14}$$

Exploiting the fact that $(\theta\widehat{\omega})$ is screw symmetric, we can easily compute the exponential of the matrix $\widehat{\omega}$ in closed form using the *Rodriguez formula*:

$$\exp(\theta\widehat{\omega}) = I + \widehat{\omega}\sin(\theta) + \widehat{\omega}^2\big(1 - \cos(\theta)\big), \tag{9.15}$$

where only the square of the matrix $\widehat{\omega}$ and sine and cosine of real numbers have to be computed. Note that this formula allows us to reconstruct the rotation matrix from the angle θ and the axis of rotation ω by simple operations and this is probably the main justification of using the axis–angle representation at all.

9.1.4.2 Exponential Maps for Rigid Body Motions

The exponential map formulation can be extended to represent rigid body motions, namely any motion composed by a rotation $\mathbf{R}$ and a translation $\mathbf{t}$. This is done by extending the parameters $\theta\omega$ with $\theta v \in \mathbb{R}^3$ which is related to the translation along the axis of rotation and the location of the axis. These six parameters form the *twist* *coordinates* $\theta\xi = \theta(v_1, v_2, v_3, \omega_1, \omega_2, \omega_3)$ of a twist. Analogous to (9.9), any rigid motion $\mathbf{G} \in \mathbb{R}^{4\times 4}$ can be written in exponential form as

$$\mathbf{G}(\theta, \omega) = \begin{bmatrix} \mathbf{R}_{3\times 3} & \mathbf{t}_{3\times 1} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} = \exp(\theta\widehat{\xi}), \tag{9.16}$$

where the 4×4 matrix $\theta\widehat{\xi} \in se(3)$ is the *twist action* and is a generalization of the screw symmetric matrix $\theta\widehat{\omega}$ of (9.10). The twist action is constructed from the twist coordinates $\theta\xi \in \mathbb{R}^6$ using the wedge operator $\wedge$

$$[\theta\xi]^\wedge = \theta\widehat{\xi} = \theta \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & v_2 \\ -\omega_2 & \omega_1 & 0 & v_3 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{9.17}$$

and its exponential can be computed using the formula

$$\exp(\theta\widehat{\xi}) = \begin{bmatrix} \exp(\theta\widehat{\omega}) & (I - \exp(\theta\widehat{\omega}))(\omega \times v + \omega\omega^T v\theta) \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{9.18}$$

with $\exp(\theta\widehat{\omega})$ computed by using the Rodriguez formula (9.15) as explained before.

9.1.4.3 The Logarithm

For human tracking it is sometimes needed to obtain the twist parameters given a transformation matrix $\mathbf{G}$. In particular if we want to obtain the resulting twist of two consecutive twists this operation is needed. In [34], a constructive way is given to compute the twist which generates a given transformation matrix $\mathbf{G}$. For the case $\mathbf{R} = \mathbf{I}$, the twist is given by

$$\theta\xi = \theta\left(0, 0, 0, \frac{\mathbf{t}}{\|\mathbf{t}\|}\right), \quad \theta = \|\mathbf{t}\|. \tag{9.19}$$

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

$$\theta = \cos^{-1}\left[\frac{tr(\mathbf{R}) - 1}{2}\right], \qquad \omega = \frac{1}{2\sin(\theta)} \begin{bmatrix} \mathbf{R}_{32} - \mathbf{R}_{23} \\ \mathbf{R}_{13} - \mathbf{R}_{31} \\ \mathbf{R}_{21} - \mathbf{R}_{12} \end{bmatrix}. \tag{9.20}$$

From (9.18) it follows that to obtain v, the matrix

$$\mathbf{A} = \big(I - \exp(\theta\widehat{\omega})\big)\widehat{\omega} + \omega\omega^T\theta \tag{9.21}$$

needs to be inverted and multiplied with the translation vector $\mathbf{t}$,

$$v = \mathbf{A}^{-1}\mathbf{t}. \tag{9.22}$$

We call this transformation from a rigid motion $\mathbf{G} \in SE(3)$ to a twist $\theta\xi \in \mathbb{R}^6$ the logarithm, $\theta\xi = \log(\mathbf{G})$.

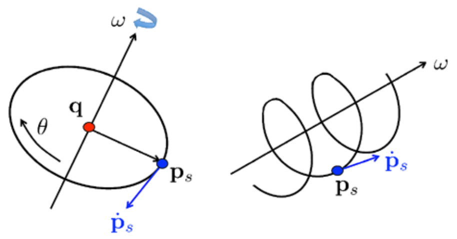

Fig. 9.2 Screw motion, *left*: the cross product of the scaled axis $\theta\omega$ and the vector $(\mathbf{p}_s - \mathbf{q})$ results in the tangential velocity of the point $\dot{\mathbf{p}}_s = \theta\omega \times (\mathbf{p}_s - \mathbf{q})$. Equivalently, the tangential velocity may be written using the twist $\dot{\mathbf{p}}_s = \widehat{\xi}\mathbf{p}_s$. *Right*: generalized screw motion with rotation and translation along the axis

9.1.4.4 Adjoint Transformation

Given a twist $\xi_b = (v_b, \omega_b) \in \mathbb{R}^6$ with coordinates in the body frame B, we can find the coordinates of the twist in the spatial frame S. Given that the configuration of B relative to frame S is the rigid motion $\mathbf{g} = (\mathbf{R}, \mathbf{t}) \in SE(3)$, the twist coordinates in the spatial frame are given by

$$\xi_s = \mathrm{Ad}_{\mathbf{g}}\,\xi_b, \quad \mathrm{Ad}_{\mathbf{g}} = \begin{bmatrix} \mathbf{R} & \mathbf{t}^\wedge\mathbf{R} \\ \mathbf{0}_{3\times 3} & \mathbf{R} \end{bmatrix}, \tag{9.23}$$

where Ad_g is the *adjoint transformation* associated with $\mathbf{g}$, see [34]. To see this, note that the angular components are related by the rotation $\omega_s = \mathbf{R}\omega_b$ (the same way we rotate points we can rotate the axes). From this it follows that $v_s = \mathbf{R}v_b + \mathbf{t}^\wedge\mathbf{R}\omega_b$. Equivalently, the action $\widehat{\xi}_s$ of a twist with twist coordinates ξ_s is related to the action $\widehat{\xi}_b$ with twist coordinates ξ_b by

$$\widehat{\xi}_s = \mathbf{G}\,\widehat{\xi}_b\,\mathbf{G}^{-1}. \tag{9.24}$$

Recall that the product of a twist $\widehat{\xi}$ by a point results in the velocity of the point $\mathbf{v}_\mathbf{p}$, see Fig. 9.2. Furthermore, *a twist with twist coordinates ξ_a in a given frame A, applies on points $\mathbf{p}_a$ defined in the same frame A and this results in the velocity of the point relative to frame A $\mathbf{v}_{\mathbf{p}_a}$*. Thus, we can interpret (9.24) the following way: the velocity in spatial coordinates of a point $\mathbf{p}_s$ is obtained by first transforming the point to body coordinates $\mathbf{p}_s \mapsto \mathbf{p}_b$ with $\mathbf{G}^{-1}$, then finding the velocity of the point in body coordinates $\mathbf{v}_{\mathbf{p}_b}$ using the twist action $\widehat{\xi}_b$ and finally transforming the velocity back to spatial coordinates $\mathbf{v}_{\mathbf{p}_s}$ with $\mathbf{G}$. One can prove that indeed this results in the spatial velocity

$$\bar{\mathbf{v}}_{\mathbf{p}_s} = \left(\mathbf{G}\,\widehat{\xi}_b\,\mathbf{G}^{-1}\right)\bar{\mathbf{p}}_s = \left(\mathbf{G}\,\widehat{\xi}_b\,\mathbf{G}^{-1}\right)\mathbf{G}\,\bar{\mathbf{p}}_b = \mathbf{G}\,\widehat{\xi}_b\,\bar{\mathbf{p}}_b = \mathbf{G}\,\bar{\mathbf{v}}_{\mathbf{p}_b}, \tag{9.25}$$

where $\bar{\mathbf{v}}_{\mathbf{p}_s} = [\mathbf{v}_{\mathbf{p}_s} 0]$ are the homogeneous coordinates of the vector $\mathbf{v}_{\mathbf{p}_s}$. An interesting property that stems from (9.24) and (9.25) is that $\mathbf{G}$ can be shifted inside the exponential

$$\exp\left(\widehat{\xi}_s\right) = \mathbf{G}\exp\left(\widehat{\xi}_b\right)\mathbf{G}^{-1} = \exp\left(\mathbf{G}\,\widehat{\xi}_b\,\mathbf{G}^{-1}\right) \tag{9.26}$$

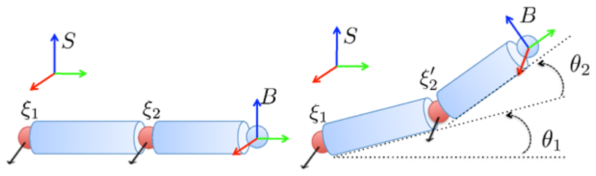

Fig. 9.3 Kinematic chain: the motion is given by the concatenation of joint angular motions. Note how the twists ξ are transformed to ξ' when parent articulations move

which means that to express a rigid body motion $\exp(\xi)$ in another coordinate system we can simply transform the corresponding twist action with $\mathbf{G}$. The same way we can interpret a rigid motion applied to a point as a coordinate transformation or as a relative motion, we can interpret the adjoint transform applied to twists as a transformation that brings a twist from their initial coordinates ξ to their coordinates ξ' (defined in the same frame) after the rigid motion $\mathbf{g}$ is applied, see Fig. 9.3. Indeed, we will make frequent use of this interpretation in the next sections when we have to keep track of human joints locations and orientations during tracking.

9.1.5 Kinematic Chains

Human motion is articulated and we want to model the motion taking all the joints into account at the same time. For example, consider the motion of the hand, this motion will be the concatenation of motions of their parent joints, wrist, elbow, shoulder and root. To formulate this we now define two coordinate frames, the spatial frame S, which is usually fixed and the body frame B, which is the coordinate system attached to the segment of interest. Note that the body frame moves along with the segment and therefore a control point in the segment in body coordinates $\mathbf{p}_b$ is always constant. Consider the arm in Fig. 9.3 with two segments and only two degrees of freedom. To obtain the coordinates of a control point in the hand in spatial coordinates $\mathbf{p}_s$ from their body coordinates $\mathbf{p}_b$ we can concatenate the rigid motions along the chain:

$$\bar{\mathbf{p}}_s = \mathbf{G}_{sb}\bar{\mathbf{p}}_b = \mathbf{G}_1\mathbf{G}_2\mathbf{G}_{sb}(\mathbf{0})\bar{\mathbf{p}}_b, \tag{9.27}$$

where $\mathbf{G}_1$, $\mathbf{G}_2$ are the rigid motion matrices of the upper and lower arm, respectively, and $\mathbf{G}_{sb}(\mathbf{0})$ is the transformation between B and S at the zero pose. By using the fact that the motion of each individual joint is generated by a twist associated with a joint axis (see Fig. 9.3) we can write the spatial coordinates of a point in the body as a function of the joint angles in the chain:

$$\bar{\mathbf{p}}_s = \mathbf{G}_{sb}(\theta_1, \theta_2) = e^{\widehat{\xi}_1\theta_1} e^{\widehat{\xi}_2\theta_2}\mathbf{G}_{sb}(\mathbf{0})\bar{\mathbf{p}}_b. \tag{9.28}$$

Any new articulation joint will represent an additional twist in the chain, Fig. 9.3. If we generalize this procedure for any limb of the human body we can define what is known in the robotics literature as the forward kinematics map. The forward kinematics is then defined as the mapping between the vector of joint angles

$\Theta = (\theta_1, \theta_2, \dots, \theta_n)^T$ to the transformation matrix between the spatial and body frames $\mathbf{G}_{sb}$. If we define Q as the space of joint angle vectors, then the forward kinematics $\mathbf{G}_{sb} : Q \to SE(3)$ is given by

$$\mathbf{G}_{sb}(\Theta) = e^{\widehat{\xi}_1 \theta_1} e^{\widehat{\xi}_2 \theta_2} \cdots e^{\widehat{\xi}_n \theta_n} \mathbf{G}_{sb}(\mathbf{0}), \tag{9.29}$$

where ξ are constant twists in the reference configuration, i.e., the starting *zero pose*. For human tracking it is usual to take $\mathbf{G}_{sb}(\mathbf{0})$ to be the identity, i.e., the body and spatial frame are coincident at the beginning for every single limb on the human body.

9.1.5.1 The Articulated Jacobian

The articulated Jacobian[1] is a matrix $\mathbf{J}_\Theta \in \mathbb{R}^{6\times n}$ that maps joint velocities to a rigid body motion velocity represented by a twist and it may be written as

$$\mathbf{J}_\Theta = \begin{bmatrix} \xi_1 & \xi_2' & \cdots & \xi_n' \end{bmatrix}, \tag{9.30}$$

where $\xi_i' = \mathrm{Ad}_{(e^{\widehat{\xi}_1 \theta_1} \dots e^{\widehat{\xi}_{i-1} \theta_{i-1}})} \xi_i$ is the ith joint twist transformed to the current pose, Fig. 9.3. To obtain ξ_i' an option is to update at every time step the twists with the accumulated motion of parent joints in the chain. Note that the form of the Jacobian is different for every limb in the body since different body parts are influenced by different joints. Now given a pose determined by Θ and point in the body in spatial coordinates $\mathbf{p}_s$ we can obtain the increment $\Delta\mathbf{p}_s$ in position as a function of the increment in parameter space $\Delta\Theta$ as

$$\Delta\bar{\mathbf{p}}_s = [\mathbf{J}_\Theta \cdot \Delta\Theta]^\wedge \bar{\mathbf{p}}_s = \left[\xi_1 \Delta\theta_1 + \xi_2' \Delta\theta_2 + \cdots + \xi_n' \Delta\theta_n\right]^\wedge \bar{\mathbf{p}}_s, \tag{9.31}$$

where $^\wedge$ is the wedge operator defined in (9.17) and we drop the homogeneous component after the multiplication of $[\mathbf{J}_\Theta \cdot \Delta\Theta]^\wedge \bar{\mathbf{p}}_s$. We can interpret the formula as follows: the total displacement of the point $\mathbf{p}_s$ is the sum of individual displacements generated by the angle increments $\Delta\theta_i$ in upper joints keeping the others fixed. It is very important to note that the result of $[\mathbf{J}_\Theta \cdot \Delta\Theta]$ are the twist coordinates ξ_s of the rotational ω and "linear velocity" v of the body expressed in the spatial frame. Note that the product $\widehat{\xi}_s \bar{\mathbf{p}}_s$ results in the point increment in homogeneous coordinates $\Delta\bar{\mathbf{p}}_s = [\Delta\mathbf{p}_s \; 0]$. Since we are not interested in the last homogeneous component, in the following we will confuse $\Delta\bar{\mathbf{p}}_s$ with $\Delta\mathbf{p}_s$ by dropping the homogeneous component after the multiplication $\widehat{\xi}_s \mathbf{p}_s$.

[1]We call it articulated Jacobian and not *manipulator Jacobian* as in Murray et al. [34] because we find it more appropriate in this context.

Table 9.1 Table of existing joints to model human motion

Joint	DoF	Unknown parameter	Example
Root	6	$\xi = \theta[v\ \omega]^T$	All body
Ball	3	$\theta\omega$	Hips
Saddle	2	θ_1, θ_2	Wrist
Revolute	1	θ	Knee

9.1.6 Human Pose Parametrization

Now we have the necessary mathematical tools to model all the joints in the human model. We identify three kinds of joints in the human body according the *DoF*, see Table 9.1. The *root joint* that determines the overall orientation and position of the body has 6 *DoF* and can be modeled as a twist $\theta\xi$ with the six components as free parameters. *Ball joints* capable of any rotation with no translation can be efficiently modeled as twist with known joint location $\mathbf{q}$ and unknown axis of rotation $\theta\omega$ [38]. Finally, simple *revolute joints* are only capable of rotations about a fixed known axis. For revolute joints the twist is constant and is given by

$$\xi = \begin{bmatrix} -\omega \times \mathbf{q} \\ \omega \end{bmatrix} \tag{9.32}$$

and the only unknown is the rotation angle θ, see Fig. 9.2 for a geometrical interpretation. This last category is very convenient to constrain the motion of 1 *DoF* joints (e.g., the knee).

In the literature the common choice is to model the root joint with six free parameters and to model all the other joints with the concatenation of revolute joints. Thereby, a ball joint is modeled by three consecutive revolute joints, i.e., three consecutive rotations about three fixed axes. The free parameters are then the angles about each of the three axes $\theta_1, \theta_2, \theta_3$. This parametrization is very similar to the Euler angles and has the same limitations in terms of singularities, i.e., it is not free of gimbal lock. However, to keep the notation simple, in the following we assume that we parametrize ball joints as three consecutive revolute joints. For a description of the parametrization of ball joints using a single free axis of rotation we refer the reader to [38].

Therefore, the pose configuration of the human is usually encoded with a scaled twist ξ for the root joint and a vector of n joint angles for the rest of the joints. Let us denote the state vector of pose parameters at time t, as

$$\mathbf{x}_t := (\xi, \Theta), \quad \Theta := (\theta_1\theta_2\ldots\theta_n). \tag{9.33}$$

Thereby, a human pose is totally determined by a D-dimensional state vector $\mathbf{x}_t \in \mathbb{R}^D$, with $D = 6 + n$.

9.1.6.1 The Pose Jacobian

For local optimization it is necessary to know the relationship between increments in the pose parameters and increments in the position of a point in a body segment. This relationship is given by the pose Jacobian $\mathbf{J}_\mathbf{x}(\mathbf{p}_s) = \frac{\Delta \mathbf{p}_s}{\Delta \mathbf{x}}$. In this paragraph, we derive the analytical expression for the pose Jacobian. We start our derivation from the expression of the point increment of (9.31). Let us denote with $\Delta\xi = [\Delta v_1\ \Delta v_2\ \Delta v_3\ \Delta\omega_1\ \Delta\omega_2\ \Delta\omega_3]$ the relative twist corresponding to the root joint. The six coordinates of the scaled relative twist $\Delta\xi$ are now free parameters we will want to estimate. By using the identity $[u + w]^\wedge = \widehat{u} + \widehat{w}$ we can rewrite (9.31) as increments in pose parameter space

$$\begin{aligned}\Delta\mathbf{p}_s &= \left[\Delta\xi + \xi_1'\Delta\theta_1 + \cdots + \xi_n'\Delta\theta_n\right]^\wedge \bar{\mathbf{p}}_s \\ &= \widehat{\Delta\xi}\,\bar{\mathbf{p}}_s + \widehat{\xi_1'}\bar{\mathbf{p}}_s\Delta\theta_1 + \cdots + \widehat{\xi_n'}\bar{\mathbf{p}}_s\Delta\theta_n,\end{aligned} \tag{9.34}$$

where we can isolate the parameters of the root joint $\Delta\xi$ rewriting $\widehat{\Delta\xi}\,\bar{\mathbf{p}}_s$

$$\widehat{\Delta\xi}\,\bar{\mathbf{p}}_s = \Delta v + \Delta\omega \times \mathbf{p}_s = \Delta v - \mathbf{p}_s^\wedge \Delta\omega = \left[\mathbf{I}_{[3\times3]} \,|\, -\mathbf{p}_s^\wedge\right]\Delta\xi \tag{9.35}$$

and substituting this expression in (9.34) again

$$\begin{aligned}\Delta\mathbf{p}_s &= \left[\mathbf{I}_{[3\times3]} \,|\, -\mathbf{p}_s^\wedge\right]\Delta\xi + \widehat{\xi_2'}\bar{\mathbf{p}}_s\Delta\theta_2 + \cdots + \widehat{\xi_n'}\bar{\mathbf{p}}_s\Delta\theta_n \\ &= \mathbf{J}_\mathbf{x}(\mathbf{p}_s)\Delta\mathbf{x},\end{aligned} \tag{9.36}$$

where $\Delta\mathbf{x} = [\Delta\xi\ \Delta\Theta]$ is the differential vector of pose parameters and

$$\mathbf{J}_\mathbf{x}(\mathbf{p}_s) = \left[\mathbf{I}_{[3\times3]} \quad -\mathbf{p}_s^\wedge \quad \widehat{\xi_1}\bar{\mathbf{p}}_s \quad \widehat{\xi_2'}\bar{\mathbf{p}}_s \quad \cdots \quad \widehat{\xi_n'}\bar{\mathbf{p}}_s\right] \tag{9.37}$$

is the positional Jacobian $\mathbf{J}_\mathbf{x}(\mathbf{p}_s) \in \mathbb{R}^{3\times D}$ of a point $\mathbf{p}_s$ with respect to the pose parameters which we denote as *pose Jacobian*. For a given point in the body $\mathbf{p}_s$ in a configuration $\mathbf{x}$, $\mathbf{J}_\mathbf{x}(\mathbf{p}_s) : \mathbb{R}^D \mapsto \mathbb{R}^3$ maps an increment of the pose parameters $\Delta\mathbf{x}$ to a positional increment of the point $\Delta\mathbf{p}_s$. We identify two main blocks in the pose Jacobian: the first six columns that correspond to the non constant relative twist $\Delta\xi$ of the root joint, and the rest of the columns (*joint columns*) that correspond to the point velocity contribution of each joint angle. Consequently, the column entries of joints that are not parents of the point are simply zero $\mathbf{0}_{3\times1}$. The analytical pose Jacobian derived here is general and will appear in every local optimization method using the parametrization described in (9.33).

9.2 Model Creation

A very important step in the pose estimation pipeline is the 3D model creation. This involves the initialization of the 3D surface mesh and the skeletal kinematic structure. We can roughly classify the approaches for shape initialization according to the level of detail. We find three main classes, methods that approximate the human body using *geometric primitives*, methods that use a subject specific *body scan* to build a 3D mesh model and finally methods that estimate *detailed shape from images* without a body scan of the subject.

9.2.1 Geometric Primitives

A wide variety of geometric primitives have been used to approximate the body shape. Early works used a simplified body model-based on a collection of articulated planar rectangles [28]. More sophisticated models have used cylinders [42], truncated cones, ellipsoids [45] or Gaussian blobs [36]. These geometric primitives can then be parametrized using very few numbers e.g., the shape of the cylinders is encoded as the height and radius. Thereby, if not initialized manually, the vector of shape parameters ϕ is estimated from images in a calibration step. The parameters include internal proportions, limb lengths and volumes.

9.2.2 Detailed Body Scans

Whole-body 3D scans provide a very accurate measurement of the surface shape. However, the model creation from a 3D scan is more involved than using simple geometric primitives. The output from a 3D scans is usually a dense 3D point cloud and a triangulated mesh. However, the triangulated mesh contains holes due to self occlusions. To initialize the model for tracking three main pre-processing steps are needed, (i) fit a template mesh to the 3D point cloud of the scanner, (ii) create a skeleton and (iii) bind skin to the skeleton bones. The last is known as skinning and the whole process is known as rigging.

- *Template mesh registration*: Since the triangulated mesh from the laser scan contains holes, a template mesh has to be morphed to fit the point cloud. This can be done with standard non-rigid registration techniques [1, 49]. Current non-rigid registration techniques require a set corresponding control points between the template mesh and the scan. The correspondences can be obtained, for example, with *Correlated Correspondence* technique which matches similar looking surface regions while minimizing the deformation [4]. Given the correspondences non-rigid registration is used to fit the template to the scan.
- *Skeleton fitting*: The skeleton determines the motion of the model. For the creation of the skeleton we must choose the number of joint articulations and the

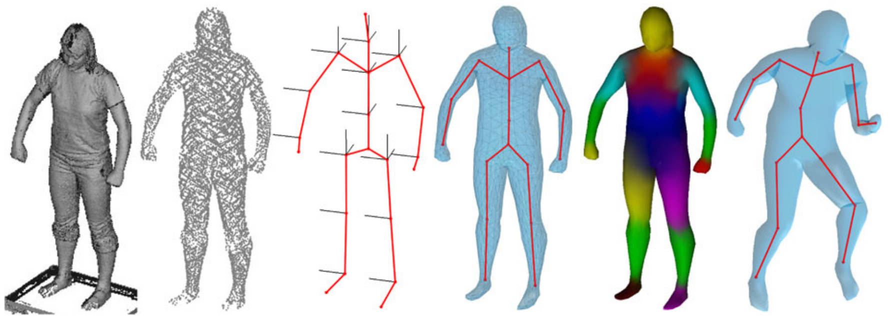

Fig. 9.4 Processing pipeline for model rigging from a body scan; from *left to right*: body scan surface, down-sampled 3D point cloud, skeleton with the twist axis orientations in black, registered template mesh, skinned model and animated model

degrees of freedom for every joint. Rough human models use only 15 degrees of freedom while models used for movie production contain over 60. For human pose estimation from images many researchers use 20–30 *DoF*, which gives a good compromise between degree of realism and robustness. For every joint we must determine two things: the location and the orientation of the axis of rotation ω, see Fig. 9.4. The skeleton is usually edited manually before tracking.

- *Skinning*: Given the registered template mesh and the skeleton we have to determine for every vertex in the surface to which body part it belongs, i.e. we must assign a joint index to every vertex. For realistic animations, however, the representation of human motion as rigid parts is too simplistic, specially for regions close to the articulations. To obtain a smooth deformation an option is to use *linear blend skinning* [32], which approximates the motion of points close to a joint by a weighted linear combination of neighboring joints. For example the motion of the shoulder vertices would be given by a combination of the torso and arm motions. The motion of a point $\mathbf{p}_s(0)$ in the reference pose is then given by

$$\bar{\mathbf{p}}_s = \sum_{i \in \mathcal{N}} w_i \mathbf{G}^i_{sb}(\mathbf{x}_t)\bar{\mathbf{p}}_s(0), \tag{9.38}$$

  where $i \in \mathcal{N}$ are the indices of their neighboring joints, and w_i are the weights. A simple rule to set the weights is to make them inversely proportional to the distance to neighboring joint locations $w_i = 1/d_i$. However, this produces severe artifacts. Several algorithms from the graphics community attempt to solve the skinning problem. As a matter of fact, open source software is available to compute the weights given a mesh and a skeleton [6]. Nevertheless, to keep notation simple, throughout this chapter we assume each point is assigned a single joint with weight equal one. We want to emphasize, however, that linear blend skinning *does not change* the formulation on kinematic chains described in the previous section since it is based on linear combinations of rigid motions.

The whole pipeline for mesh registration and rigging is shown in Fig. 9.4.

9.2.3 Detailed Shape from Images

Body scan models are limited by the availability of range scanners. To overcome this limitation a recent research direction has focused on the estimation of detailed shape from images [5, 26]. This is achieved by finding parametrization learned from a database of human scans that encodes human shape and pose variation across individuals [2, 3]. All subjects in the database are scanned in different poses to account for both shape and pose deformation. The pose is usually encoded by a combination of rigid and non-rigid deformations, and the shape variation is modeled with a set of PCA coefficients learned from the database.

As a final comment, there exist approaches that use neither a skeleton nor shape knowledge from a database [12, 18]. In contrast, such approaches directly deform the mesh geometry by non-rigid deformation to fit a set of multiview silhouettes. While impressive results are achieved with such methods, high quality silhouettes are needed and at least eight cameras are used.

9.3 Optimization

Now that we have the mathematical tools to generate the 3D models and to represent human motion, our aim is to recover this motion form one or multiview images. Model-based algorithms are classified as generative model approaches because independently of the optimization scheme used, they all model the likelihood of the observations for a given configuration of pose parameters. The pose that best explains the observations is typically found by minimizing an error function that measures how well the model fits the image data. Even using multiple cameras and relatively simple background settings this poses a hard optimization problem. Difficulties arise from model-image matching ambiguities, depth ambiguities and the high dimensionality of the state space. An additional difficulty is that the space of plausible poses only represents a small region of the full parameter space $\mathbb{R}^D$. The ability to obtain better results by constraining search to a sub-space of plausible poses will be discussed at length in Chap. 10. The key components for successful tracking, which we will describe here, are the design of the cost function and the optimization strategy. In this section we describe the different optimization strategies for human pose estimation and the type of error functions used.

9.3.1 Local Optimization

Given an initial estimate, local optimization methods are based on iteratively linearizing the error function to find a descent direction. Usually, these methods converge to a local optimum and consequently their performance strongly depends on the initialization. During tracking, the knowledge of the estimates in previous frames can simplify the task: in the simplest case the initial estimate is given by the pose obtained in the previous frame, or alternatively motion models can be used to make good predictions closer to the true pose. We distinguish three main families of local optimization methods for human pose estimation: methods based on *correspondences*, *optical flow* and *regions*.

9.3.1.1 Correspondence-Based

Almost all early approaches for 3D human pose estimation were *correspondence-based* and still it remains one of the most popular strategies. A reason for that is that these approaches are computationally efficient while providing very accurate results in many situations. The key idea is to collect a set of correspondences between 3D points $\mathbf{p}_i$ of the model and the *image observations* $\mathbf{r}_i$. Then, the distance between the projection of the 3D model points $\tilde{\mathbf{r}}_i$ (*predictions*) and the image observations is minimized with respect to the pose parameters $\mathbf{x}_t$, see Fig. 9.5.

Hence, *correspondence-based* algorithms consist of three main stages

1. *Feature extraction*: extract image observations (e.g., silhouettes, edges, SIFT features)
2. *Model image association*: match the model 3D points with the image observations and collect this correspondences
3. *Descent strategy*: find the pose parameters that bring the model points into correspondence with the image observations

Feature extraction: Different features like image silhouettes, image edges, and SIFT have been used and combined for human pose estimation. Edge and silhouettes where used in very early works and continue to be dominant for human pose estimation because they are relatively easy to extract and are stable to illumination changes. Therefore, we will explain in detail a motion capture system based on silhouettes and then the integration of additional features like SIFT will become obvious. In the context of human tracking a silhouette is a binary image with white pixels indicating the foreground i.e., the region of the subject we want to track. In indoor environments, silhouettes can be obtained with great accuracy via background subtraction techniques [35]. In outdoor environments, it is considerably more challenging but also possible if background images are available. Once the silhouettes are obtained, the image contour is obtained with an edge detector.

Model image association: For the correspondences, since we want to predict the image contour, only points belonging to the *occluding contour* are considered. A point belongs to the occluding contour $\mathbf{p}_i \in \mathscr{O}$ if its surface normal $\hat{\mathbf{n}}$ is perpendicular to the line $\mathbf{L}$ connecting the camera center $\mathbf{O}$ and the point. In other words,

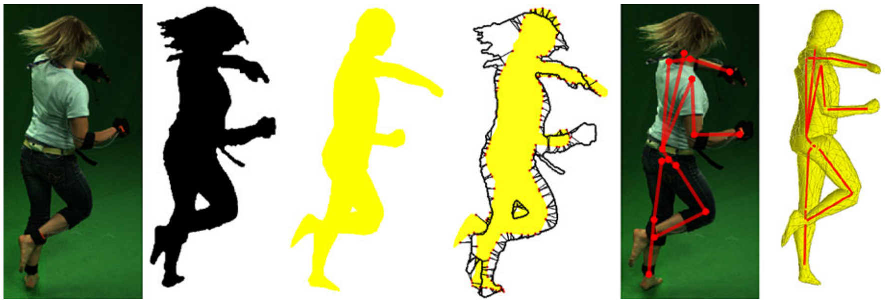

Fig. 9.5 From *left to right*: original image, silhouette from background subtraction, rendering of the projected mesh at the current pose, correspondences by contour matching, animated model seen from a virtual view

the occluding contour is the set of points in the mesh that project to the silhouette contour of the projected mesh.

To find the points of the occluding contour there are two main strategies, the first and the simplest one is to test for every point in the mesh if the surface normal is perpendicular to the projection ray:

$$\mathbf{p} \in \mathscr{O} \quad \text{if } \hat{\mathbf{n}} \cdot (\mathbf{p} - \mathbf{C}) < \varepsilon, \tag{9.39}$$

where ε is a small positive threshold. In practice, however, this approach is problematic since the accuracy of $\hat{\mathbf{n}}$ strongly depends on the mesh resolution (number of vertices of the mesh). One solution is to look for sign changes of the angle between the triangle normal and the projection ray. The second strategy is to first render a binary silhouette projection on the image, project all the vertices of the mesh and retain only those on the silhouette boundary. To render the silhouette image, all the visible surface triangles are projected to the image and filled using typical graphics raster-scan algorithms. Alternatively, the rendering can be very efficiently performed on the GPU using OpenGL. At this point we have two sets of points, the 3D points in the occluding contour $\mathbf{p}_i \in \mathscr{O}$ and the 2D points from the image contour $\mathbf{r}_i \in \mathscr{I}$. The correspondences can be found by finding for every point projection $\tilde{\mathbf{r}}_i = \text{Pr}_c(\mathbf{p}_i)$ the k-nearest neighbors in the image contour. This will result in a set of 3D–2D correspondences $(\mathbf{p}_i, \mathbf{r}_i)$ or 2D–2D correspondences $(\tilde{\mathbf{r}}_i, \mathbf{r}_i)$. We note that finding correct correspondences is a hard problem with probably many local minima. To leverage this, additional terms based on overlap between the model and image silhouette can be included into the matching cost [44].

Descent strategies: Collecting many of these correspondences $(\tilde{\mathbf{r}}_i, \mathbf{r}_i)$ the error function $e : \mathbb{R}^D \mapsto \mathbb{R}$ may be defined as the sum of squared re-projection errors in the image

$$e(\mathbf{x}_t) = \sum_i^N \mathbf{e}_i^2(\mathbf{x}_t) = \sum_i^N \left\| \tilde{\mathbf{r}}_i(\mathbf{x}_t) - \mathbf{r}_i \right\|^2 \tag{9.40}$$

which we want to minimize with respect to the pose parameters $\mathbf{x}_t$. Note that in the case of 2D–2D correspondences the individual residual errors $\mathbf{e}_i \in \mathbb{R}^2$ are 2D error vectors $\mathbf{e}_i = (\Delta r_{i,x}, \Delta r_{i,y})$. Equation (9.40) is a classical non-linear least squares that can be re-written in vector form as

$$e(\mathbf{x}_t) = \mathbf{e}^T \mathbf{e}, \tag{9.41}$$

where $\mathbf{e} \in \mathbb{R}^{2N}$ is the total residual error $\mathbf{e} = (\mathbf{e}_1^T, \mathbf{e}_2^T, \dots, \mathbf{e}_N^T)$. Equation (9.41) can be efficiently minimized using a *Gauss–Newton* style minimization. The trick is to iteratively linearize the vector function $\mathbf{e} \in \mathbb{R}^{2N}$ around the solution with the Jacobian matrix $\mathbf{J}_t$ to find a descent step. In the literature, the expression for the Jacobian matrix is often omitted due to space limitations. Therefore, we reproduce here how to derive the analytical expression of the Jacobian matrix $\mathbf{J}_t \in \mathbb{R}^{2N \times D}$ of the residual error $\mathbf{e}$. We start by deriving the expression for the Jacobian of the error of a single correspondence $\mathbf{J}_{t,i} = \frac{\Delta \mathbf{e}_i}{\Delta \mathbf{x}}$. It is straightforward to see that the individual error Jacobian equals the prediction Jacobian $\mathbf{J}_{t,i} = \frac{\Delta \mathbf{e}_i}{\Delta \mathbf{x}} = \frac{\Delta \tilde{\mathbf{r}}_i}{\Delta \mathbf{x}}$ because only the prediction $\tilde{\mathbf{r}}_i$ depends on the pose parameters. Recall that the matrix $\mathbf{J}_{t,i} \in \mathbb{R}^{2 \times D}$ maps increments in the pose parameters $\Delta \mathbf{x}$ to increments in the predictions $\Delta \tilde{\mathbf{r}}_i$. To compute the Jacobian it is useful to identify the set of transformations applied to a point $\mathbf{p}_s(0)$ in the reference pose to the final projection in the image $\tilde{\mathbf{r}}$. We can visualize this with the diagram

$$\begin{tikzcd} \mathbf{p}_s(0) \arrow[r, "\mathbf{G}_{sb}(\mathbf{x}_t)"] & \mathbf{p}_s \arrow[r, "g_c := \mathbf{M}_{ext}"] & \mathbf{p}_c \arrow[r, "g_p := f(\frac{X}{Z} + o_x, \frac{X}{Z} + o_y)"] & \tilde{\mathbf{r}}, \end{tikzcd}$$

where $\mathbf{G}_{sb}(\mathbf{x}_t)$ is the concatenation of rigid motions in the kinematic chain given by the pose parameters $\mathbf{x}_t$, $g_c(\mathbf{p}) \mapsto \mathbf{M}_{ext}\mathbf{p}$ is the extrinsic camera matrix that transforms a point from spatial coordinates $\mathbf{p}_s$ to camera coordinates $\mathbf{p}_c$ and $g_p(X, Y, Z) \mapsto (f\frac{X}{Z} + o_x, f\frac{X}{Z} + o_y)$ is the perspective projection of 3D point in camera coordinates onto the image plane (with f denoting the focal length, (o_x, o_y) the principal point and we assume the skew coefficient is one). Now we can compute the Jacobians $\mathbf{J}_c \in \mathbb{R}^{3\times 3}$, $\mathbf{J}_p \in \mathbb{R}^{2\times 3}$ of the functions g_c, g_p separately as

$$g_c : \mathbb{R}^3 \to \mathbb{R}^3, \quad \mathbf{p}_c^i = \mathbf{M}_{ext}\bar{\mathbf{p}}_s^i = \mathbf{R}_{cs}\mathbf{p}_s^i + \mathbf{t}_{cs} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad \mathbf{J}_c = \mathbf{R}_{cs},$$

$$g_p : \mathbb{R}^3 \to \mathbb{R}^2, \ \tilde{\mathbf{r}}_i = g_p(\mathbf{p}_c^i) = \left(f\frac{X}{Z} + o_x, f\frac{Y}{Z} + o_y \right), \ \mathbf{J}_p = f \begin{bmatrix} \frac{1}{Z} & 0 & -\frac{X}{Z^2} \\ 0 & \frac{1}{Z} & -\frac{Y}{Z^2} \end{bmatrix},$$

where the Jacobian of g_c is directly the rotational component of the extrinsics, $\mathbf{R}_{cs}$ because it is a linear map and the Jacobian of g_p is computed by direct application of the definition of the Jacobian matrix. By applying the chain rule, the Jacobian of the composed mapping $\mathbf{J}_{t,i}$ might be written as

$$\mathbf{J}_{t,i} = \frac{\Delta \tilde{\mathbf{r}}_i}{\Delta \mathbf{x}_t} = \frac{\Delta \tilde{\mathbf{r}}_i}{\Delta \mathbf{p}_c} \cdot \frac{\Delta \mathbf{p}_c}{\Delta \mathbf{p}_s} \cdot \frac{\Delta \mathbf{p}_s}{\Delta \mathbf{x}_t} = \mathbf{J}_p \mathbf{R}_{cs} \mathbf{J}_{\mathbf{x}}(\mathbf{p}_s^i), \tag{9.42}$$

where $\mathbf{J}_\mathbf{x}(\mathbf{p}_s^i)$ is the pose Jacobian derived earlier in (9.37). It is straightforward to see that the Jacobian of total residual error $\mathbf{J}_t \in \mathbb{R}^{2N \times D}$ may be written by stacking the individual point Jacobians $\mathbf{J}_{t,i} \in \mathbb{R}^{2 \times D}$

$$\mathbf{J}_t = \frac{\Delta \mathbf{e}}{\Delta X_t} = \begin{bmatrix} \mathbf{J}_{t,1} \\ \mathbf{J}_{t,2} \\ \vdots \\ \mathbf{J}_{t,N} \end{bmatrix}. \tag{9.43}$$

With the analytical expression of the residual Jacobian the Gauss–Newton method calculates the descent step as follows:

$$\begin{aligned} \Delta\mathbf{x} &= \arg\min_{\Delta\mathbf{x}} \frac{1}{2}\mathbf{e}^T(\mathbf{x}_t + \Delta\mathbf{x})\mathbf{e}(\mathbf{x}_t + \Delta\mathbf{x}) \\ &= \arg\min_{\Delta\mathbf{x}} \frac{1}{2}(\mathbf{e} + \mathbf{J}_t\Delta\mathbf{x})^T(\mathbf{e} + \mathbf{J}_t\Delta\mathbf{x}) \\ &= \arg\min_{\Delta\mathbf{x}} \frac{1}{2}\mathbf{e}^T\mathbf{e} + \Delta\mathbf{x}^T\mathbf{J}_t^T\mathbf{e} + \frac{1}{2}\Delta\mathbf{x}^T\mathbf{J}_t^T\mathbf{J}_t\Delta\mathbf{x}, \end{aligned} \tag{9.44}$$

where $\mathbf{e} = \mathbf{e}(\mathbf{x}_t)^T$ and $\mathbf{J}_t = \mathbf{J}_t(\mathbf{x}_t)$ are evaluated at the current estimation $\mathbf{x}_t$. Finally, derivating with respect to $\Delta\mathbf{x}$ and equating to zero we find that the descent step is

$$\Delta\mathbf{x} = -\left(\mathbf{J}_t^T\mathbf{J}_t\right)^{-1}\mathbf{J}_t^T\mathbf{e}. \tag{9.45}$$

At every iteration of the Gauss–Newton algorithm the step is computed using (9.45) and the pose parameters are updated $\mathbf{x}_{t+1} = \mathbf{x}_t + \Delta\mathbf{x}$. This procedure is repeated until the algorithm converges. The step $\Delta\mathbf{x}$ always decreases the error function $e(\mathbf{x}_t)$ as long as the Jacobian matrix $\mathbf{J}_t$ has full rank. In the *Levenberg–Marquadt* algorithm, the Gauss–Newton step is modified

$$\Delta\mathbf{x} = -\left(\mathbf{J}_t^T\mathbf{J}_t + \mu\mathbf{I}\right)^{-1}\mathbf{J}_t^T\mathbf{e} \tag{9.46}$$

by introducing an additional dynamically chosen parameter $\mu\mathbf{I}$ that improves the performance. If the step decreases the error, the step is accepted and the value of μ is reduced. If the step increases the error, μ is increased and a new step is computed. When μ is large the method performs like standard gradient descent, slow but guaranteed to converge. When μ is small it performs like Gauss–Newton. Once the algorithm has converged the obtained pose estimate is used as initialization for the next frame, new correspondences are found in the new image and the process is repeated. For large motions one often needs to re-project the model to the image several times to obtain refined correspondences, similar to the standard Iterative Closest Point (ICP) registration method [7, 52].

Different error functions: Different error functions have been proposed in the literature. For example, Rosenhahn et al. [39] directly minimize the sum of squared

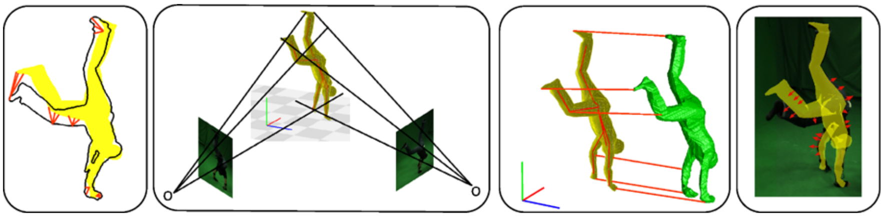

Fig. 9.6 Different error functions, from *left to right*: minimization of re-projection error (2D–2D correspondences), point to line distance error minimization (2D–3D correspondences), point to point distance error minimization (3D–3D correspondences) and region-based tracker

distances between 3D model points $\mathbf{p}_s^i$ and the projection rays L_i casted by the corresponding 2D image observations $\mathbf{r}_i$. Writing the projection line in Plücker coordinates $L_i = (\mathbf{m}_i, \mathbf{n}_i)$ the error may be written as

$$e(\mathbf{x}_t) = \sum_i^N \mathbf{e}_i^2(\mathbf{x}_t) = \sum_i^N \left\| \mathbf{p}_s^i(\mathbf{x}_t) \times \mathbf{n}_i - \mathbf{m}_i \right\|^2, \tag{9.47}$$

where the residuals are 3D distance error vectors $\mathbf{e}_i \in \mathbb{R}^3$. In this case it is straightforward to show that the Jacobian of the error of one correspondence pair is given by $\mathbf{J}_{t,i} = \mathbf{n}_i^\wedge \mathbf{J}_\mathbf{x}(\mathbf{p}_s^i) \in \mathbb{R}^{3\times D}$.

Another alternative used by several authors [14, 16] is to first reconstruct the visual hull [30] from the multiview images obtaining a rough volume of the human shape. Then the matching is done between the model points and the points from the visual hull resulting in a set of 3D–3D correspondences $(\mathbf{p}_s^i, \mathbf{q}_s^i)$. The error function is then simply the distance between 3D points

$$e(\mathbf{x}_t) = \sum_i^N \mathbf{e}_i^2(\mathbf{x}_t) = \sum_i^N \left\| \mathbf{p}_s^i(\mathbf{x}_t) - \mathbf{q}_s^i \right\|^2 \tag{9.48}$$

with $\mathbf{e}_i \in \mathbb{R}^3$ where in this case the Jacobian is directly $\mathbf{J}_{t,i} = \mathbf{J}_\mathbf{x}(\mathbf{p}_s^i)$.

In Fig. 9.6 an illustration of common error functions is shown.

Combining features: Other kind of features like SIFT or optical flow can be integrated as additional correspondences into this framework as long as they can be predicted given a pose estimate. The combination of features should robustify the tracker [11].

9.3.1.2 Optical Flow-Based

Optical flow is the apparent motion in the image projection of 3D object in the world. The displacement $[u\ \ v]$ of every pixel $[x\ \ y]$ from one frame to the next one is computed assuming that the intensity remains constant. This is known as the *brightness constancy* assumption and it may be written as $I(x, y, t-1) = I(x+u, y+v, t)$. The first-order Taylor expansion of the right hand side of the equation leads to the remarkable normal optical flow constraint equation [33],

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \cdot \begin{bmatrix} u \\ v \end{bmatrix} - I_t = 0, \tag{9.49}$$

where I_x, I_y and I_t are the spatial and temporal derivatives of the image. Generally, neighboring pixels are assumed to move together according to a given motion model. Bregler et al. [9, 10] used a human motion model to parameterize the motion flow $[u \; v]$. Specifically, he finds for every pixel in the image the corresponding 3D point in the human model $\mathbf{p}_s$. Then the motion $[u \; v]$ is simply given by the projection of the 3D point displacement $\Delta\mathbf{p}_s$ onto the image plane. This can be written as

$$\begin{bmatrix} I_x & I_y \end{bmatrix} \cdot \mathrm{Pr}_c(\Delta\mathbf{p}_s) - I_t = 0, \tag{9.50}$$

where we recall that Pr_c denotes projection on the image plane (in [9, 10] an orthographic camera projection is assumed), and the point displacement is given by $\Delta\mathbf{p}_s = \mathbf{J}_\mathbf{x}\Delta\mathbf{x}_t$ (9.36). Note the strong correlation with correspondence-based algorithms by interpreting (9.50) as the linearization of the error function $\mathbf{e}_i(\mathbf{x}_t) = I(x+u, y+v, t) - I(x, y)$ for one correspondence. Here only $[u \; v]$ depend on the pose parameters. The total error function $e(\mathbf{x}_t)$ can be interpreted then as the sum of squared pixel intensity differences. Unfortunately, approaches relying exclusively on optical flow have two main drawbacks. First, when the motion is large, the Taylor expansion in (9.49) produces large estimate errors. Second, while image features like edges and silhouettes provide an absolute measurement, relying on optical flow causes error accumulation which results in drift and tracking failure [31]. Nevertheless, [10] was the first work in human pose estimation to use the elegant twists and exponential maps formulation from the robotics community.

9.3.1.3 Region-Based

Region-based methods are based on separating the foreground figure (the human silhouette) from the background by maximizing the dissimilarity between region density functions (usually the density functions are approximated by simple histograms of pixel intensities and colors). Popular approaches to achieve this are level sets or graph-cuts [8, 13]. This process can be coupled with human pose estimation in an EM scheme. An initial human pose estimate defines two regions in the image, namely the interior of the projected silhouette and the exterior. This initial boundary is then used as a shape prior for a level-set segmentation. Thereafter, correspondences between the segmentation and the projected silhouette are obtained and the pose is estimated using a correspondence-based method. This process of pose estimation and segmentation is iterated until convergence. Some works have coupled the feature extraction and the descent strategy. The work by [17, 40] skips the segmentation step and directly shifts the points belonging to the occluding contour

$\tilde{\mathbf{r}} \in \mathscr{O}$ inwards or outwards (orthogonal to the contour line) according to the posterior probability densities. If the foreground posterior is bigger than the background posterior the point is shifted outwards, otherwise the point is shifted inwards, see Fig. 9.6. This implicitly generates correspondence pairs between points and shifted points which feed a correspondence-based tracker.

9.3.1.4 Probabilistic Interpretation

We have seen that local optimization methods, either correspondence-based, optical flow or region-based are based on defining an error function and linearizing the residual error vector via its Jacobian to find a descent step. As a final comment, we note that one can give a probabilistic interpretation to the error functions defined above. Gathering image observations at time t in a random vector $\mathbf{y}_t$, (observations can be 2D point locations, lines, 3D points, feature descriptors, appearance, ...) the MAP estimate is given by

$$\mathbf{x}_{t,\mathrm{MAP}} = \arg\max_{\mathbf{x}_t} p(\mathbf{x}_t|\mathbf{y}_t) = \arg\max_{\mathbf{x}_t} p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t), \tag{9.51}$$

where $p(\mathbf{y}_t|\mathbf{x}_t)$ is the likelihood of the observations for a given pose $\mathbf{x}_t$ and $p(\mathbf{x}_t)$ is the prior knowledge we have about human motion (which will be discussed in the next chapter). If the errors associated with the observations are independent and have a Gaussian distribution, the likelihood takes the form

$$p(\mathbf{x}_t|\mathbf{y}_t) = p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t) \propto \exp\left(-\sum_i^N \mathbf{e}_i^2\left(\mathbf{y}_t^i|\mathbf{x}_t\right)\right)p(\mathbf{x}_t), \tag{9.52}$$

where $\mathbf{e}_i^2(\mathbf{y}_t^i|\mathbf{x}_t)$ is the individual error for a given image observation. Equivalently to (9.51), the MAP estimate can be obtained by the minimization of the negative log-likelihood and an additional prior term $e_p(\mathbf{x}_t)$

$$\mathbf{x}_{\mathrm{MAP}} = \arg\min_{\mathbf{x}_t} -\log\bigl(p(\mathbf{y}_t|\mathbf{x}_t)\bigr) - \log\bigl(p(\mathbf{x}_t)\bigr) = \arg\min_{\mathbf{x}_t} e(\mathbf{x}_t) + e_p(\mathbf{x}_t). \tag{9.53}$$

Therefore, the minimization of the sum of squared error functions $e(\mathbf{x})$ defined in the previous subsections (e.g. (9.40), (9.47) and (9.48)) are equivalent to a MAP estimator if the observation errors are independent and Gaussian (without prior it is actually equivalent to a maximum likelihood (ML) estimator). Consequently, the error function should be designed to model the cost density associated with the observations [45]. This interpretation will be particularly useful when we see optimization methods based on stochastic search. Probabilistic formulation of the pose tracking problem is more thoroughly discussed in Sect. 10.1.1 of Chap. 10.

9.3.2 Particle-Based Optimization and Filtering

A well known problem of local optimization methods is that since they are based on propagating a single pose hypothesis, when there is a tracking error the system can in general not recover from it. To overcome this limitation, stochastic search techniques have been introduced for human pose estimation. This group of methods are based on approximating the likelihood of the image given the pose parameters by propagating a set of particles from one time step to the next one.

9.3.2.1 Particle Filter

Problems in human pose estimation arise from kinematic singularities, depth and orientation ambiguities and occlusions. For all these reasons the posterior density $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ and the observation process $p(\mathbf{y}_t|\mathbf{x}_t)$ are highly peaked and highly multimodal. The image likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$ is the probability of observing certain image features given a pose $\mathbf{x}_t$, and $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ is the probability of the pose parameters considering the history of all observations from previous images $1:t$. To see this multimodality, note that many configurations in pose parameter space $\mathbf{x}$ explain well the observations $\mathbf{y}_{1:t}$ (for example any rotation by an angle α about the axis of a limb will project to almost the same image location). It is well known that in this case a Kalman filter will fail. In these cases the posterior can be approximated by a particle filter.[2] A particle filter approximates the posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ by a set of particles $\{\pi_t^{(i)}, \mathbf{x}_t^{(i)}\}_{i=1}^N$ where the weights are normalized so that $\sum_N \pi_t^{(i)} = 1$. Each particle $\mathbf{x}_t^{(i)}$ corresponds to one configuration of pose parameters (9.33), and the weights are chosen to be proportional to the likelihood $\pi^{(i)} \propto p(\mathbf{y}_t|\mathbf{x}_t = \mathbf{x}_t^{(i)})$. At each time step the pose parameters can be estimated by the mean of the weighted particles,

$$\mathbf{x}_t^* = \mathbb{E}_\mathbf{x}[\mathbf{x}_t] \simeq \sum_N \pi_t^{(i)} \mathbf{x}_t^{(i)} \tag{9.54}$$

or by the mode of the particle set $\mathbf{x}_t^* = \mathbb{M}_\mathbf{x}[\mathbf{x}_t] = \mathbf{x}_t^{(i)}$ with $\pi_t^{(i)} = \max(\pi_t^{(n)})$.

Assuming a first-order Markov process ($p(\mathbf{x}_t|\mathbf{x}_{1:t-1}) = p(\mathbf{x}_t|\mathbf{x}_{t-1})$) the posterior distribution can be updated recursively:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) \propto p(\mathbf{y}_t|\mathbf{x}_t)p(\mathbf{x}_t) \int p(\mathbf{x}_t|\mathbf{x}_{t-1})p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})\,d\mathbf{x}_{t-1}, \tag{9.55}$$

where the integral computes the pose prediction from the previous time step posterior $p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1})$ propagated with the dynamical model $p(\mathbf{x}_t|\mathbf{x}_{t-1})$. The prediction is then weighted by the likelihood function $p(\mathbf{y}_t|\mathbf{x}_t)$ times the prior $p(\mathbf{x}_t)$ if available. In a particle filter setting, (9.55) is approximated by *importance sampling*.

[2]See also Chap. I, Sect. 6.2.2.

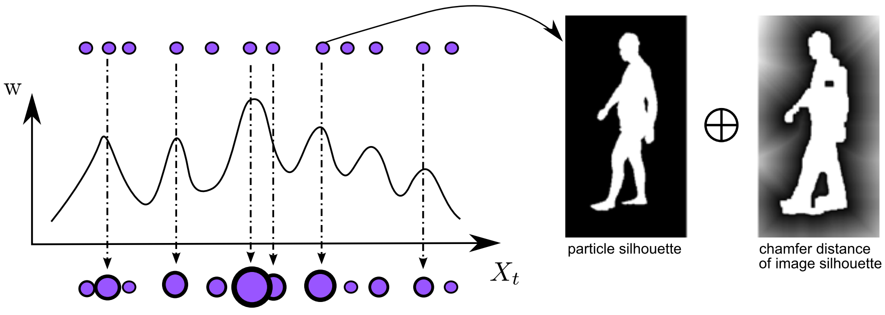

Fig. 9.7 *Particle Filter*: on the *left* the weighting function is shown as a function of the pose parameters. The function is multimodal and it is difficult to tell where the maximum is from the particle distribution. On the *right*: the weighting function is evaluated for every particle $\mathbf{x}_t^{(i)}$. The weighting function should be fast to evaluate, in this example, it consists of a simple overlap measure between the particle silhouette and the chamfer distance transformed image silhouette

Given a set of weighted particles approximating the posterior in the previous frame $\mathscr{P}_{t-1}^+ := \{\pi_{t-1}^{(i)}, \mathbf{x}_{t-1}^{(i)}\}_{i=1}^N$, N particles are drawn from $\mathscr{P}_{t-1}$ with replacement and probability proportional to their weights obtaining a new set of unweighted particles $\{\tilde{\mathbf{x}}_{t-1}^{(i)}\}_{i=1}^N$. Thereafter, the particles are propagated to the next frame by sampling from the dynamical model $p(\mathbf{x}_t|\tilde{\mathbf{x}}_{t-1}^{(i)})$ producing a new unweighted set of predictions $\mathscr{P}_t^- := \{\mathbf{x}_t^{(i)}\}_{i=1}^N$. Finally, every particle in $\mathscr{P}_t^-$ is weighted according to the likelihood in the new frame $\pi_t^{(i)} = p(\mathbf{y}_t|\mathbf{x}_t^{(i)})$ obtaining the final weighted set $\mathscr{P}_t^+ := \{\pi_t^{(i)}, \mathbf{x}_t^{(i)}\}_{i=1}^N$ that approximates the updated posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$.

Likelihood functions: Ideally, to model the observation process $p(\mathbf{y}_t|\mathbf{x}_t)$ the complicated image formation process has to be synthesized, i.e., illumination, human appearance rendering on the image, clothing, occlusions, shadows, etc. Since $p(\mathbf{y}_t|\mathbf{x}_t)$ has to be evaluated for every particle in the set $p(\mathbf{y}_t|\mathbf{x}_t = \mathbf{x}_t^{(i)})$, this is obviously computationally infeasible. In practice, to make the problem tractable an intuitive function $w(\mathbf{y}_t, \mathbf{x}_t = \mathbf{x}_t^{(i)})$ is constructed that approximates the probability $p(\mathbf{y}_t|\mathbf{x} = \mathbf{x}_k^{(i)})$. This function takes into account only image observations $\mathbf{y}_t$ that can be modeled easy and efficiently (e.g., edges, coarse foreground appearance or silhouettes). Actually, the error functions $e(\mathbf{x}_t)$ described for local optimization might be used to set the weights w as

$$w\big(\mathbf{y}_t, \mathbf{x}_t = \mathbf{x}_t^{(i)}\big) = \exp\big(-e\big(\mathbf{x}_t^{(i)}\big)\big), \tag{9.56}$$

where, as explained in Sect. 9.3.1.4, we interpret the error as the cost density associated with the observations. To gain in efficiency, a chamfer distance can be pre-computed in the original image silhouette or edge map, see Fig. 9.7. Then, simple overlap measures between the synthesized particle silhouette and the chamfer distance image are computed [19, 23, 51]. Another commonly used feature in the weighting function is appearance, whose associated cost can be evaluated with histogram comparison [23, 42]. For a comparative study of the influence of different likelihood/weighting functions we refer the reader to [43]. Nonetheless, the computation of $w(\mathbf{y}_t, \mathbf{x}_t = \mathbf{x}_t^{(i)})$ is a very expensive operation if it has to be evaluated for many particles. In addition, the number of particles required to approximate $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ grows exponentially with the dimensionality of $\mathbf{x}$, which makes the particle filter intractable for realistic human models with more than 30 *DoF*. Furthermore, even using a large number of particles the search can be misdirected if many local modes are present in $w(\mathbf{y}_t, \mathbf{x}_t = \mathbf{x}_t^{(i)})$; see Fig. 9.7.

9.3.2.2 Annealed Particle Filter

The annealed particle filter (APF) is a modification of the particle filter and it was introduced for human pose estimation by Deutscher et al. [19]. The goal here is to modify the particle filter such that the number of needed particles is drastically reduced and the particles do not congregate around local maxima. The APF is motivated from simulated annealing methods designed for global optimization of multimodal functions [29]. The key idea is to gradually introduce the influence of narrow peaks in the weighting function $w(\mathbf{y}_t, \mathbf{x}_t)$. This is achieved by starting a search run in successive layers gradually changing the weighting function as

$$w_m(\mathbf{y}_t, \mathbf{x}_t) = w(\mathbf{y}_t, \mathbf{x}_t)^{\beta m}, \tag{9.57}$$

for $\beta_0 > \beta_1 > \cdots > \beta_M$. The run is started at layer M, where w_M is broad and reflects the overall trend of w without the influence of so many local maxima. As we move to the next layer, β increases and therefore the local peaks become more accentuated. For initialization, an initial set of samples is drawn from a proposal distribution q_{M+1}. During tracking we might choose this distribution to be the set of particles from the previous frame or the propagated particles if we use a motion model. For initialization in the first frame the distribution should be spread with a high variance, see for example [22]. Once the algorithm is initialized the optimization consists of the following steps executed at every layer: *weighting*, *resampling* and *diffusion*, see Fig. 9.8 and the pseudo code in Algorithm 1.

- *Weighting*: The surviving particles of the previous layer are assigned new weights $w_{t,m}(\mathbf{y}_t, \mathbf{x}_{t,m}^{(i)})^{\beta m}$ with the new annealing scheme βm. At this point a new proposal distribution has been formed $q_m = \{\pi_{t,m}^{(i)}, \mathbf{x}_{t,m}^{(i)}\}_{i=1}^N$.
- *Resampling*: N new samples are drawn from the distribution q_m with probability equal to the weights, this can be efficiently done with multinomial sampling. Note that particles with high weight (low error) will be selected with higher probability and therefore a higher number of particles concentrate around the maximum of the weighting function. Gall et al. [21, 23] proposes a generalization of the resampling strategy that improves the performance of the APF. In this modification, particles with high weight are with high probability retained in the set without replacement.

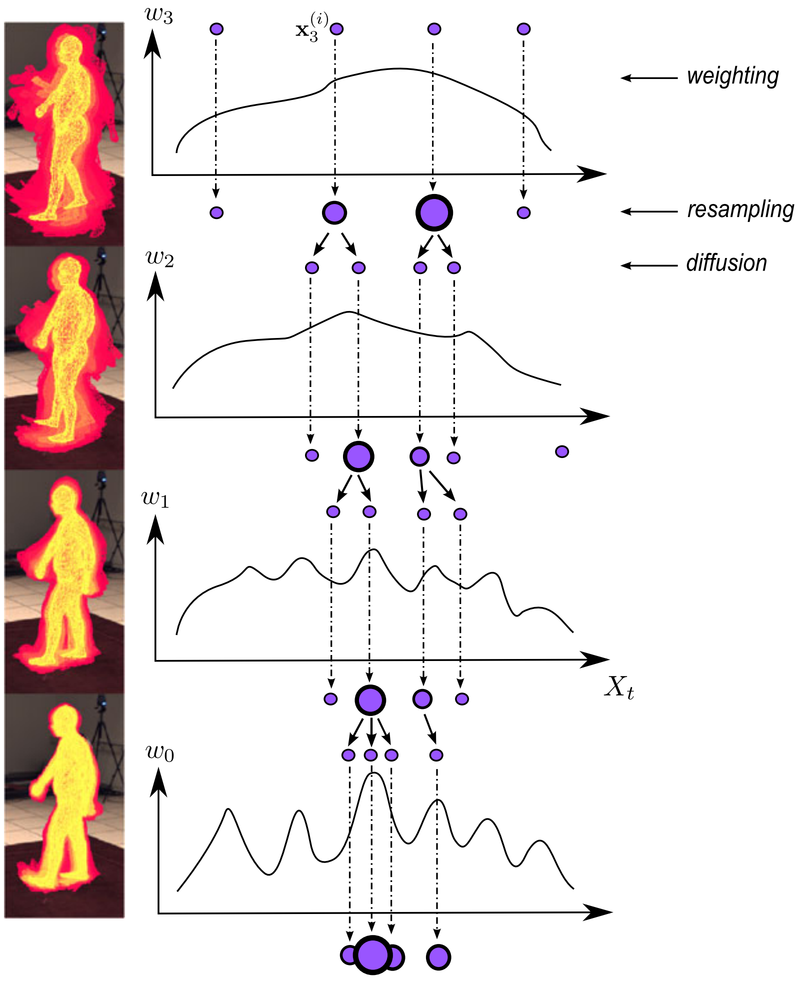

Fig. 9.8 *Annealed particle filter*: At each layer the weighting, resampling and diffusion steps are performed. The influence of the peaks is gradually introduced into the weighting function w_m. Consequently, the particles converge at the global maximum in the last layer w_0 and do not get trapped in local maxima peaks as opposed to the standard particle filter. Additionally, on the left column, the distribution of the particles projected to the image is shown, where particles with higher weights are brighter (left column images are courtesy of Gall et al. [23])

- *Diffusion*: In order to explore the search space the particles are diffused by some function. A common choice is to shift every particle by adding a sample from a multivariate Gaussian with covariance Σ_m. The covariance Σ_m can be chosen to be proportional to the particle set variance since this provides a measure of uncertainty (The more uncertainty, the farther away we have to explore the search space.) The run in the layer terminates with the diffusion step and the particles are used to initialize the weighting step of the next layer.

Algorithm 1 Annealed particle filter

```
Require: number of layers $M$, number of samples $N$, initial distribution $q_{M+1}$,
  Initialize: Draw $N$ initial samples from $q \to \mathbf{x}_{t,m}^{(i)}$
  for layer $m = M$ to $0$ do
    1. WEIGHTING
    start from the set of unweighted particles of the previous layer
    for $i = 1$ to $N$ do
      compute $w_{t,m}(\mathbf{y}_t, \mathbf{x}_{t,m}^{(i)})^{\beta m}$
      set $\pi_{t,m}^{(i)} \propto w_{t,m}(\mathbf{y}_t, \mathbf{x}_{t,m}^{(i)})^{\beta m}$
    end for
    set $q_m = \{\pi_{t,m}^{(i)}, \mathbf{x}_{t,m}^{(i)}\}_{i=1}^{N}$
    2. RESAMPLING
    draw $N$ samples from $q_{t,m} \to \mathbf{x}_{t,m}^{(i)}$        {Can be done with multinomial sampling}
    3. DIFFUSION
    $\mathbf{x}_{t,m-1}^{(i)} = \mathbf{x}_{t,m}^{(i)} + \mathbf{B}_{t,m}$        {$\mathbf{B}_{t,m}$ is a sample from a multivariate Gaussian with $\mathcal{N}(0, \Sigma)$}
  end for
```

Once the algorithm has finished the last annealing run, the pose estimate is obtained by the mean of the final set of particles of the last layer, $\mathbf{x}_t^* = \mathbb{E}_{\mathbf{x}_t}[q_0] = \sum_i \pi_{t,0}^{(i)} \mathbf{x}_{t,0}^{(i)}$. Although the APF can be used recursively as the standard PF in (9.55) using some heuristics, it can be considered a single frame global optimization algorithm. While the APF is computationally more efficient in locating global minima in the likelihood function $p(\mathbf{y}_t|\mathbf{x}_t)$ than the particle filter, the main disadvantage is not being able to work within a robust Bayesian framework [19]. The reason for that is that the set of particles of q_0 congregate *only* around one maximum of the observation process $p(\mathbf{y}_t|\mathbf{x}_t)$ at the current frame t. Therefore, the particles are not well distributed for the next frame and usually a heuristic has to be used to spread them to search in the next frame. By contrast, in the PF (which needs much more samples) the final particle set represents the total posterior $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ which might contain multiple maxima. Therefore the PF can represent inherent ambiguities by maintaining multiple modes at the expense of a significant loss in accuracy.

9.3.2.3 Tailored Optimization Methods

Although an exhaustive survey of the proposed sampling methods and likelihood functions used in the literature for human tracking is out of the scope of this chapter, it is worth to highlight some optimization procedures tailored for the problem of human pose estimation. Choo and Fleet [15] use a Markov Chain similar to (9.55) but use every particle as initialization for local optimization on the likelihood function. Similarly, Sminchisescu and Triggs [45] also combine particle-based sampling with local optimization, but additionally sample along the most uncertain directions calculated from the local likelihood Hessian matrix at the fitted optima. Along this lines, importance samplers have been proposed that focus samples on regions likely to contain transitions states leading to nearby minima [46, 47]. Another way to locate multiple optima in monocular images is to exploit the geometry of the problem to deterministically enumerate the set of poses that result in the same projection [48, 50]. To make the sampling tractable in the high-dimensional space, state partitions have also been used [20, 24]. These partitions are either specified manually by sequentially tracking different kinematic branches (e.g. torso, limbs and head) [24] or selected automatically from the parameter variances in the particle set [20]. Every optimization scheme has its own advantages/disadvantages but a common key component in all of them for successful tracking is a good background subtraction.

9.4 Discussion

The basic mathematical tools necessary for anyone who wants to implement a human pose estimation system have been introduced, namely kinematic structure representation and model creation. A unified formulation has been presented for the most common model-based pose estimation algorithms seen in the literature. We have seen that the model image association problem for pose estimation is usually formulated as the minimization/maximization of an error/likelihood function. The two main strategies have been described, namely local and particle-based optimization. Local optimization methods are faster and more accurate but in practice, if there are visual ambiguities, or really fast motions, the tracker might fail catastrophically. To achieve more robustness, particle filters can be used because they can represent uncertainty through a rigorous Bayesian paradigm. The problem here is that the number of particles needed to achieve reasonable results grows exponentially with the dimensionality of the pose parameter space which makes them unpractical for human models with more than 20 *DoF*. To reduce the number of needed particles, the annealed particle filter can be used at the expense of not being able to work in a fully Bayesian framework. A further problem of global optimization methods is that while robust, they do not provide a single temporally consistent motion but rather a jittery motion which must be post-processed to obtain visually acceptable results. Combinations of global and local optimization have also been proposed to compensate for the drawbacks of each strategy [15, 23, 45]. Nonetheless, current approaches do not capture detailed movement such as hand orientation or axial arm rotation. This stems from depth and orientation ambiguities inherent in the images (i.e., any rotation about a limb axis projects to the same image). To overcome this limitations Pons-Moll et al. [37] proposes a hybrid tracker that combines correspondence-based local optimization with five inertial sensors placed at body extremities to obtain a much accurate and detailed human tracker, see Fig. 9.9. The key idea is to fuse both sensor types in a joint optimization to exploit the advantages of each sensor type (accurate position from images and accurate orientation from inertial sensors).

Although over the last decade significant progress has been made in model-based human pose estimation, there remain a number of open problems and challenges.

First, while tracking of walking motions in semi-controlled settings is more or less reliable, robust tracking of arbitrary and highly dynamic motions is still challenging even in controlled setups. Second, monocular tracking is a highly ambiguous and difficult problem which is still far from being solved. Although monocular

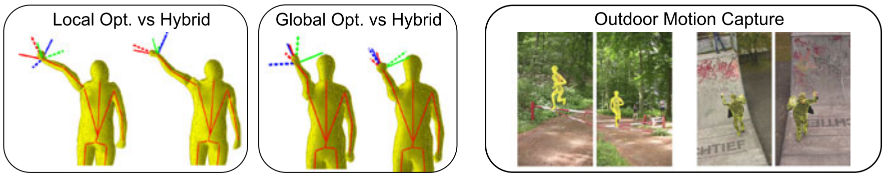

Fig. 9.9 *Left*: current approaches, either local or global optimization cannot capture axial rotation accurately; the combination of a video-based tracker with inertial sensors (hybrid tracker) allows to capture detailed motion. *Right*: motion capture in uncontrolled scenarios is one of the principal future challenges (outdoor images on the right are courtesy of Hasler et al. [27])

tracking has attracted a lot of attention because of the complexity of the problem, most solutions rely on restrictive setups with very simple motions. Monocular tracking is particularly interesting because in many practical settings only a single view sequence is available as in video footage archives such as YouTube. Third, tracking arbitrary motions in outdoor settings has been mostly unaddressed [27] Fig. 9.9, and remains one of the open problems in computer vision. Indeed, this must be one of the final goals since in these scenarios standard commercial motion capture systems based on optical markers cannot be used. Outdoor human tracking would allow to capture sport motions in their real competitive setup. Fourth, tracking people in the office or in the streets interacting with the environment is still an extremely challenging problem to be solved. In this chapter we have covered the mathematical tools for generative model-based pose estimation. Another active research area to increase robustness and accuracy is the modeling of priors for human motion; this involves using temporal context, building motion and physical models which will be described in the next chapter.

References

1. Allen, B., Curless, B., Popović, Z.: Articulated body deformation from range scan data. In: ACM Transactions on Graphics, pp. 612–619. ACM, New York (2002) [151]
2. Allen, B., Curless, B., Popović, Z.: The space of human body shapes: Reconstruction and parameterization from range scans. In: ACM Transactions on Graphics, pp. 587–594. ACM, New York (2003) [153]
3. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. ACM Trans. Graph. **24**, 408–416 (2005) [153]
4. Anguelov, D., Srinivasan, P., Pang, H.C., Koller, D., Thrun, S., Davis, J.: The correlated correspondence algorithm for unsupervised registration of nonrigid surfaces. In: Advances in Neural Information Processing Systems, p. 33. MIT Press, Cambridge (2005) [151]
5. Balan, A.O., Sigal, L., Black, M.J., Davis, J.E., Haussecker, H.W.: Detailed human shape and pose from images. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2007) [153]
6. Baran, I., Popović, J.: Automatic rigging and animation of 3d characters. In: ACM Transactions on Graphics, p. 72. ACM, New York (2007) [152]
7. Besl, P., McKay, N.: A method for registration of 3d shapes. IEEE Trans. Pattern Anal. Mach. Intell. **12**, 239–256 (1992) [157]

8. Boykov, Y., Veksler, O., Zabih, R.: Fast approximate energy minimization via graph cuts. IEEE Trans. Pattern Anal. Mach. Intell. **23**(11), 1222–1239 (2001) [159]
9. Bregler, C., Malik, J.: Tracking people with twists and exponential maps. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 8–15 (1998) [159]
10. Bregler, C., Malik, J., Pullen, K.: Twist based acquisition and tracking of animal and human kinematics. Int. J. Comput. Vis. **56**, 179–194 (2004) [159]
11. Brox, T., Rosenhahn, B., Gall, J., Cremers, D.: Combined region and motion-based 3d tracking of rigid and articulated objects. IEEE Trans. Pattern Anal. Mach. Intell. **32**(3), 402–415 (2010) [158]
12. Cagniart, C., Boyer, E., Ilic, S.: Free-form mesh tracking: A patch-based approach. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1339–1346 (2010) [153]
13. Chan, T.F., Vese, L.A.: Active contours without edges. IEEE Trans. Image Process. **10**(2), 266–277 (2001) [159]
14. Cheung, K.M.G., Baker, S., Kanade, T.: Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1 (2003) [158]
15. Choo, K., Fleet, D.J.: People tracking using hybrid Monte Carlo filtering. In: IEEE International Conference on Computer Vision, vol. 2, pp. 321–328 (2001) [165,166]
16. Corazza, S., Mündermann, L., Gambaretto, E., Ferrigno, G., Andriacchi, T.P.: Markerless motion capture through visual hull, articulated icp and subject specific model generation. Int. J. Comput. Vis. **87**(1), 156–169 (2010) [158]
17. Dambreville, S., Sandhu, R., Yezzi, A., Tannenbaum, A.: Robust 3d pose estimation and efficient 2d region-based segmentation from a 3d shape prior. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) European Conference on Computer Vision. Lecture Notes in Computer Science, vol. 5303, pp. 169–182. Springer, Berlin (2008) [159]
18. de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., Thrun, S.: Performance capture from sparse multi-view video. In: ACM Transactions on Graphics, pp. 1–10. ACM, New York (2008) [153]
19. Deutscher, J., Blake, A., Reid, I.: Articulated body motion capture by annealed particle filtering. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 126–133 (2000) [162,163,165]
20. Deutscher, J., Davison, A., Reid, I.: Automatic partitioning of high dimensional search spaces associated with articulated body motion capture. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 2 (2001) [166]
21. Gall, J., Potthoff, J., Schnorr, C., Rosenhahn, B., Seidel, H.: Interacting and annealing particle filters: Mathematics and a recipe for applications. J. Math. Imaging Vis. **28**, 1–18 (2007) [163]
22. Gall, J., Rosenhahn, B., Seidel, H.: Clustered stochastic optimization for object recognition and pose estimation. In: DAGM. Lecture Notes in Computer Science, vol. 4713, pp. 32–41. Springer, Berlin (2007) [163]
23. Gall, J., Rosenhahn, B., Brox, T., Seidel, H.: Optimization and filtering for human motion capture. Int. J. Comput. Vis. **87**, 75–92 (2010) [162-164,166]
24. Gavrila, D., Davis, L.: 3D model based tracking of humans in action: A multiview approach. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (1996) [166]
25. Grassia, S.: Practical parameterization of rotations using the exponential map. J. Graph. Tools **3**, 29–48 (1998) [143]
26. Hasler, N., Ackermann, H., Rosenhahn, B., Thormaehlen, T., Seidel, H.: Multilinear pose and body shape estimation of dressed subjects from image sets. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 1823–1830 (2010) [153]
27. Hasler, N., Rosenhahn, B., Thormaehlen, T., Wand, M., Gall, J., Seidel, H.-P.: Markerless motion capture with unsynchronized moving cameras. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 224–231 (2009) [167]

28. Ju, S.X., Black, M.J., Yacoob, Y.: Cardboard people: A parameterized model of articulated image motion. In: International Workshop on Automatic Face and Gesture Recognition, pp. 38–44 (1996) [151]
29. Kirkpatrick, S., Gelatt Jr., C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983) [163]
30. Laurentini, A.: The visual hull concept for silhouette-based image understanding. IEEE Trans. Pattern Anal. Mach. Intell. **16**(2), 150–162 (1994) [158]
31. Lepetit, V., Fua, P.: Monocular model-based 3d tracking of rigid objects: A survey. Found. Trends Comput. Graph. Vis. **1**(1), 1–89 (2005) [159]
32. Lewis, J.P., Cordner, M., Fong, N.: Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In: ACM Transactions on Graphics, pp. 165–172. ACM, New York (2000) [152]
33. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: International Joint Conference on Artificial Intelligence, vol. 3, pp. 674–679 (1981) [159]
34. Murray, R.M., Li, Z., Sastry, S.S.: Mathematical Introduction to Robotic Manipulation. CRC Press, Baton Rouge (1994) [140,143,145,146,148]
35. Piccardi, M.: Background subtraction techniques: A review. In: Proc. IEEE Int Systems, Man and Cybernetics Conf., vol. 4, pp. 3099–3104 (2004) [154]
36. Plankers, R., Fua, P.: Articulated soft objects for video-based body modeling. In: IEEE International Conference on Computer Vision, vol. 1, pp. 394–401 (2001) [151]
37. Pons-Moll, G., Baak, A., Helten, T., Mueller, M., Seidel, H.-P., Rosenhahn, B.: Multisensor-fusion for 3d full-body human motion capture. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 663–670 (2010) [166]
38. Pons-Moll, G., Rosenhahn, B.: Ball joints for marker-less human motion capture. In: Proc. IEEE Workshop Applications of Computer Vision (WACV) (2009) [149]
39. Rosenhahn, B., Brox, T.: Scaled motion dynamics for markerless motion capture. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2007) [157]
40. Schmaltz, C., Rosenhahn, B., Brox, T., Cremers, D., Weickert, J., Wietzke, L., Sommer, G.: Region-based pose tracking. In: Proc. 3rd Iberian Conference on Pattern Recognition and Image Analysis, vol. 4478, pp. 56–63 (2007) [159]
41. Shoemake, K.: Animating rotation with quaternion curves. ACM SIGGRAPH Computer Graphics **19**, 245–254 (1985) [143]
42. Sidenbladh, H., Black, M., Fleet, D.: Stochastic tracking of 3d human figures using 2d image motion. In: Vernon, D. (ed.) European Conference on Computer Vision. Lecture Notes in Computer Science, vol. 1843, pp. 702–718. Springer, Berlin (2000) [151,163]
43. Sigal, L., Balan, A.O., Black, M.J.: Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. Int. J. Comput. Vis. **87**(1), 4–27 (2010) [163]
44. Sminchisescu, C.: Consistency and coupling in human model likelihoods. In: International Workshop on Automatic Face and Gesture Recognition (2002) [155]
45. Sminchisescu, C., Triggs, B.: Covariance scaled sampling for monocular 3d body tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1 (2001) [151,160,165]
46. Sminchisescu, C., Triggs, B.: Building roadmaps of local minima of visual models. In: European Conference on Computer Vision, pp. 566–582 (2002) [165]
47. Sminchisescu, C., Triggs, B.: Hyperdynamics importance sampling. In: European Conference on Computer Vision, pp. 769–783 (2002) [165]
48. Sminchisescu, C., Triggs, B.: Kinematic jump processes for monocular 3d human tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2003) [166]
49. Sumner, R.W., Popović, J.: Deformation transfer for triangle meshes. In: ACM Transactions on Graphics, pp. 399–405. ACM, New York (2004) [151]

50. Taylor, C.J.: Reconstruction of articulated objects from point correspondences in a single uncalibrated image. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 677–684 (2000) [166]
51. Vondrak, M., Sigal, L., Jenkins, O.C.: Physical simulation for probabilistic motion tracking. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2008) [162]
52. Zhang, Z.: Iterative points matching for registration of free form curves and surfaces. Int. J. Comput. Vis. **13**(2), 119–152 (1994) [157]