Stochastic optimisation for high-dimensional tracking in dense range maps

M. Bray, E. Koller-Meier, P. Müller, N.N. Schraudolph and L. Van Gool

Abstract: The main challenge of tracking articulated structures like hands is their many degrees of freedom (DOFs). A realistic 3-D model of the human hand has at least 26 DOFs. The arsenal of tracking approaches that can track such structures fast and reliably is still very small. This paper proposes a tracker based on stochastic meta-descent (SMD) for optimisations in such high-dimensional state spaces. This new algorithm is based on a gradient descent approach with adaptive and parameter-specific step sizes. The SMD tracker facilitates the integration of constraints, and combined with a stochastic sampling technique, can get out of spurious local minima. Furthermore, the integration of a deformable hand model based on linear blend skinning and anthropometrical measurements reinforces the robustness of the tracker. Experiments show the efficiency of the SMD algorithm in comparison with common optimisation methods.

1 Introduction

Relatively little work has been devoted to the tracking of articulated structures [1–5]. Although the Condensation algorithm by Isard and Blake [6] has shown to be a robust and powerful stochastic sampling approach, it is not really applicable to this kind of tracking application. A dense sampling of the underlying probability distribution functions is needed to guarantee a good solution, but this soon becomes unfeasible for higher-dimensional state spaces. In order to cope with this problem, one either has to lower the dimensionality, or to devise schemes that work well with fewer samples. This paper contributes to the latter types of approach. It proposes a new type of optimisation method that is both fast and able to avoid getting trapped in spurious local minima. We demonstrate its potential for hand tracking. As input we use dense 3-D data acquired with a structured light system in which skin colour detection helps to find the hands. Before going into the details, we will review concisely the literature on the subject.

As mentioned above, one approach is to lower the dimensionality of the problem. With the use of an action specific dynamic model, Sidenbladh *et al.* [1] manage to reduce the number of parameters in the state vector. Nevertheless, they still have to process quite a large number of samples in their particle filter to track walking humans (25 DOFs) and their algorithm requires approximately 50 seconds per frame (on a 1 GHz processor). Also Wu *et al.* [2] decrease the state space dimensionality. They represent hand articulations in a lower dimensional space by a set of linear manifolds constructed from basis configurations. Taking advantage of such a representation of hand articulation, they apply a particle filter based on importance sampling techniques but still need about 100 samples.

The alternative is to devise strategies that require fewer samples. Deutscher *et al.* [3] have proposed a modified particle filter based on a simulated annealing algorithm. In contrast to lowering the dimensionality, the annealed particle filter (APF) reduces the number of samples and increases efficiency by a factor of up to 10. Moreover, APF localises the tracked object even better as it increases the chances of finding the global minimum. In order to reach a good accuracy with sparse sampling, Sminchisescu and Triggs [4] have combined global sampling with local optimisation based on a gradient descent method.

Basically, both Sminchisescu's approach and APF introduce particles with a more sophisticated behaviour that thereby get closer to becoming a tracker in their own right. Here, we will build further on a tracker that is based on stochastic meta-descent (SMD). In SMD, a cost function is minimised with a novel gradient descent method with step sizes that vary over time and between dimensions. In each iteration, the appropriate changes are determined automatically by a meta-level gradient descent. Its integration into a visual hand tracking framework needs modifications since owing to the incomplete 3-D data (holes), we do not have access to the gradient everywhere and have to handle these special cases. Moreover, we naturally incorporate constraints that are in general difficult to integrate or computationally expensive. Additionally, the introduction of some stochasticity into the evaluation of the objective function increases the chance of getting out of local minima. This new technique is also more efficient than previously known methods. A more detailed comparison between SMD and APF is presented in Section 5, which will demonstrate that APF is slower than our tracker when a similar accuracy is needed. Sminchisescu and Triggs [4] approach was also reported to be slower.

A somewhat similar approach of using gradient descent is taken by Rehg and Kanade [5]. They describe an articulated object by a set of overlapping templates and minimise the residual between the observed image and the templates by a

---

IEE Proceedings online no. 20045113

doi: 10.1049/ip-vis:20045113

Paper first received 15th July and in revised form 17th December 2004. Originally published online 5th July 2005

M. Bray, E. Koller-Meier, P. Müller and L. Van Gool are with Computer Vision Lab, ETH Zurich, Switzerland

N.N. Schraudolph is with National ICT Australia, Australian National University, Canberra, Australia

L. Van Gool is also with ESAT/PSI/Visics, KULeuven, Belgium

E-mail: bray@vision.ee.ethz.ch

standard gradient descent. In comparison, we use 3-D rather than 2-D video, and deal with more DOFs (2 fingers in [5]). Also, our gradient descent scheme is more sophisticated, a requirement also envisaged by Rehg and Kanade (p. 616 in [5]).

The cost function of our tracker is based on an underlying 3-D hand model. Recently, a strand of developments has appeared that works directly from 2-D hand templates. They create a database of different views and configurations of the target with their corresponding features such as silhouette moments [7] or feature combinations [8, 9]. Then, the goal is to explore efficiently this database to find a match to an input image. Rosales *et al.* [7] propose the use of a specialised mappings architecture (SMA), a nonlinear supervised learning method, to find the mapping from 2-D hand image features (Hu-moments) to 3-D configurations (from monocular views). Tomasi *et al.* [9] decompose the task of 3-D tracking from a video stream into 2 tasks: classification and interpolation. Tracking finger spelling is used to illustrate their approach. Stenger *et al.* [10] introduce a tree-based estimator within a Bayesian filtering framework in order to track an articulated hand. The idea relies on a multiscale discretisation of the state space where shape templates are used to compute the likelihood. Later, Stenger *et al.* [11] present a cascade of classifiers arranged in a tree so as to recognise hand poses.

This type of method requires a quickly growing amount of memory as the dimensionality of the target increases. Furthermore, it is sometimes difficult to recover the 3-D hand pose from a single extracted 2-D hand template when the viewpoint provides poor feature information or high ambiguity. Finally, some of these methods assume the availability of a 3-D hand model in order to simplify the template generation.

2 The hand model

Quite a few hand models consist of simple primitives, such as cylinders or truncated cones. We use a more detailed, deformable model, based on 3-D scans. Our model gives us a higher accuracy in the reproduction of actual hand shapes, and hence also better supports tracking by producing more subtle effects like the deformation of the palm.

Hierarchically structured character skin deformation was introduced by Catmull [12]. In subsequent work, Magnenat-Thalmann *et al.* [13] and Komatsu [14] went beyond rigid skinning techniques and demonstrated human body deformation driven by an underlying skeleton. They developed custom programmed algorithms for each joint. We have chosen this non-physical approach as it is computationally efficient. For the same reason we use polygons instead of NURBS.

2.1 Hand model

In constructing a hand model, the skeleton provides a hierarchical structure for animating the deformation of the created surface. It is defined by a series of segments (bones) with joints where the model should bend. The angle between two such segments is called the 'joint angle'. The human hand is constrained by dependencies [15] that reduce the model to 30 DOFs: 5 joint angles for each finger (3 for flexion, 1 for abduction, 1 for twist) except for the thumb, which has 4 joint angles (3 for flexion, 1 for abduction). The palm has 6 DOFs for the wrist's translation and rotation. An overview of these DOFs is given in Fig. 1. McDonald *et al.* [15] established hard constraints for the twists. Hard constraints describe absolute limitations imposed on the range of motion of the joints. To the best of our knowledge, the dynamic dependencies between the twists and the other joints have not yet been fully explored. In order to leave open the possibility for studying these dynamic dependencies, we have chosen to consider the twists as DOFs, moving in a range of ±4 degrees. The hard constraints and dependencies of the other angles are the same as used in [16]. For instance the dependencies between PIP and DIP angles (Fig. 1), namely $DIP = (2)/(3)PIP$, further reduce the model to 26 DOFs.

2.2 Linear blend skinning

In order to put a skin around the skeleton, we use linear blend skinning. Nowadays, linear blend skinning algorithms are included in most commercial 3-D modelling and animation software packages and they go by many names. Magnenat-Thalmann *et al.* [13] called it 'joint-dependent local deformation', Lewis *et al.* [17] called it 'skeleton subspace deformation', Alias' Maya calls it 'smooth skinning' [Note 1], and in one of the most recent publications in this field, it is called 'linear blend skinning' [18].

First, a hierarchical skeleton is placed inside a static polygon-model. This initial pose has, again, various names: binding pose, dress pose etc. The difference from a usual polygon model is that every vertex has blending weights as additional attributes, typically one for each 'influence', i.e. each joint. A new pose is computed by rigidly transforming every vertex according to its binding pose for all its influencing joints and, after that, linearly blending them together for every vertex.

The position of a vertex $\boldsymbol{v}_c$, influenced by n joints, for a skeletal configuration c, is computed as:

$$\boldsymbol{v}_c = \sum_{i=1}^{n} \gamma_i M_{i,c} M_{i,b}^{-1} \boldsymbol{v}_b \qquad (1)$$

where γ_i are the blending weights, $\boldsymbol{v}_b$ is the binding pose position of some vertex $\boldsymbol{v}$, $M_{i,c}$ is the global transformation matrix of the ith joint in configuration c and $M_{i,b}^{-1}$ is the inverse of the binding pose associated with the ith joint. The case $c = b$ results in the same location of $\boldsymbol{v}(\boldsymbol{v}_c = \boldsymbol{v}_b)$ as in its binding pose. Thus, the first product of (1) between $M_{i,b}^{-1}$ and $\boldsymbol{v}_b$ gives the vertex $\boldsymbol{v}_b$ in the local reference frame of the influence joint i. The second product with $M_{i,c}$ results in the vertex $\boldsymbol{v}_b$ in world coordinates with respect to the influence joint i and the configuration $\boldsymbol{c}$. This product is then weighted according to γ_i.

An illustration of the linear blend skinning computation is given in Fig. 2. The intensity in the left image is proportional to the weight of the MP joint of the index finger on the corresponding vertices of the proximal index finger phalanx. Similarly, the right image shows the weights that the PIP joint has on the vertices of the middle phalanx. Note how vertices around the PIP joint are influenced by both this joint and the MP one. In any case, the weights of all vertices sum up to one. Also, note that the resulting skin motion tangential to the hand surface is not necessarily realistic. What matters more to our application is the overall shape, a purpose for which these weights have been optimised (see Section 2.3). The two products $M_{index_{\mathbf{MP}},b}^{-1}\boldsymbol{v}_b$ and $M_{index_{\mathbf{PIP}},b}^{-1}\boldsymbol{v}_b$ are then multiplied by their respective transformation matrices from their local reference frame to the world reference frame. Finally, they are weighted and summed to yield the position of the vertex given the configuration $\boldsymbol{c}$.

Note 1: http://www.aliaswavefront.com

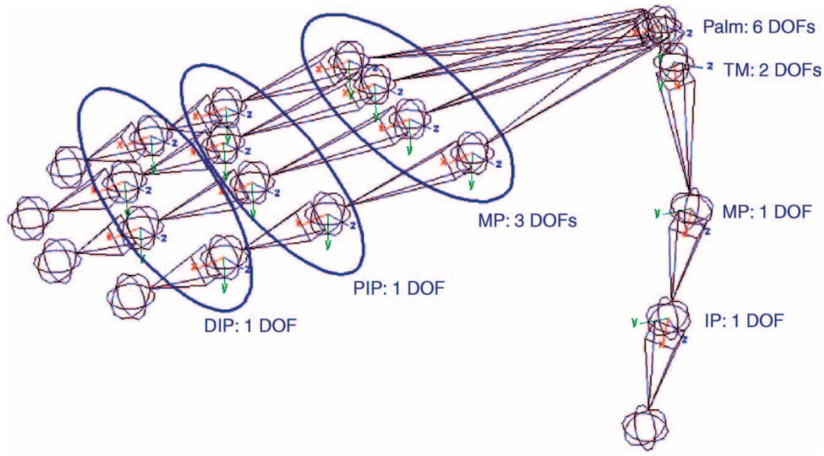

Fig. 1 *The hand model and its degrees of freedom (DOFs)*

DIP indicates the 'distal interphalangeal' joints, PIP the 'proximal interphalangeal' joints, MP the 'metacarpophalangeal' joints, TM the 'trapeziometacarpal' joint and IP the 'interphalangeal' joint

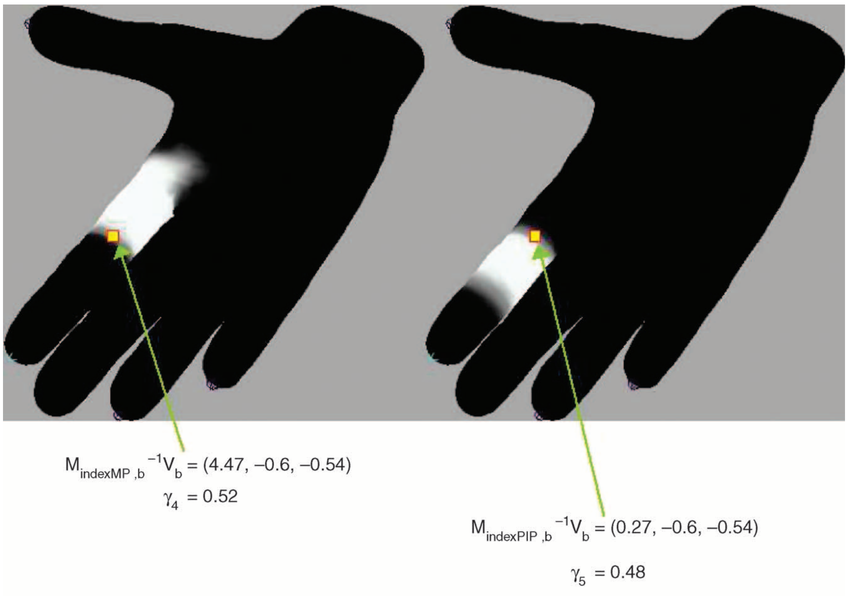

Fig. 2 *Illustration of the linear blend skinning computation*

The intensity is proportional to the weight of the proximal phalanx of the index finger on the left image. On the right, the intensity is the proportion to the weight of the middle phalanx of the index finger

2.3 Authoring the model

The polygonal surface that makes up the skin of the hand is modelled with Alias' Maya. By using amongst other things the more advanced sculpting tools, the generic model was fitted to the scanned hand of our test person. For the scanning we used a structured-light system [Note 2]. The final mesh (see Fig. 3) consists of 9051 vertices. This resolution gives a good trade-off for our application: it is accurate enough and the deformation can be solved easily in real-time on a modern personal computer.

The anthropometrical measurements in [15, 19] give us a very good approximation for joint positions. In these two papers, statistical relationships between the skin and the underlying skeleton have been established.

It is important that the skin deforms naturally as the skeleton moves. When the joints rotate, the skin bulges or indents near the joints depending on the blending weights. Authoring the weights (i.e. influences, see Section 2.2) appropriately is not straightforward. Figure 3 illustrates our weights painted onto the hand model. We used a maximum of 3 influences per vertex. As an example, the deformation of the skin between the thumb and the palm is influenced by the palm and the two first phalanges of the thumb.

While linear blend skinning is very fast to evaluate and compact in memory and data-structure, it is notorious for its authoring difficulties and its shortcomings. The famous 'shrunken elbow' problem is shown in Fig. 4 for a bent finger. As previously seen in Fig. 2, one can observe a gradient while painting the weights around the finger joints to ensure a human-like deformation. If the width of this gradient is too wide, we observe deformations similar to the one presented in the left illustration of Fig. 4. In contrast, if the width of this gradient is tighter, this results in natural skin deformations such as the one presented in the right illustration of Fig. 4. Complex deformations like wrinkles etc. cannot be represented owing to the algorithm's linear nature. This fortunately poses no problems for our tracking task.

Note 2: http://www.eyetronics.com

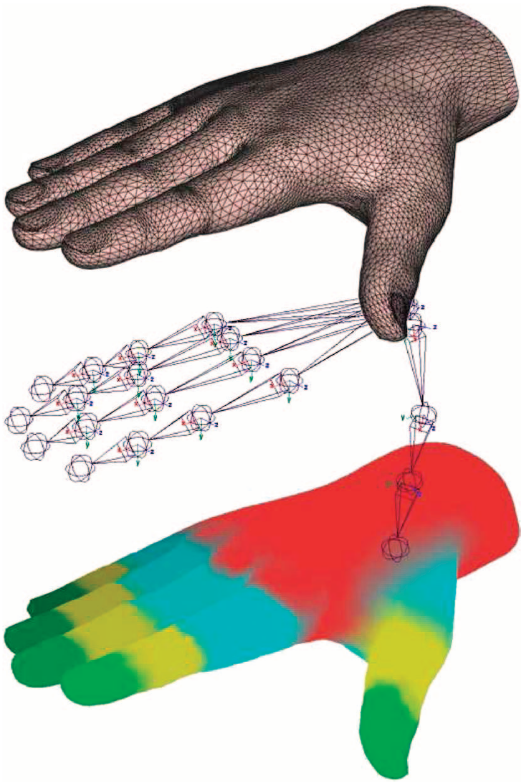

Fig. 3 *Final mesh and weights painted onto the hand model*

The hand model as polygonal surface (top). The joints in the skeleton (middle) define the locations where the hand should bend. The intensity encoded weights connecting the surface to the skeleton are shown at the bottom. Each intensity represents the influence area of its corresponding joint

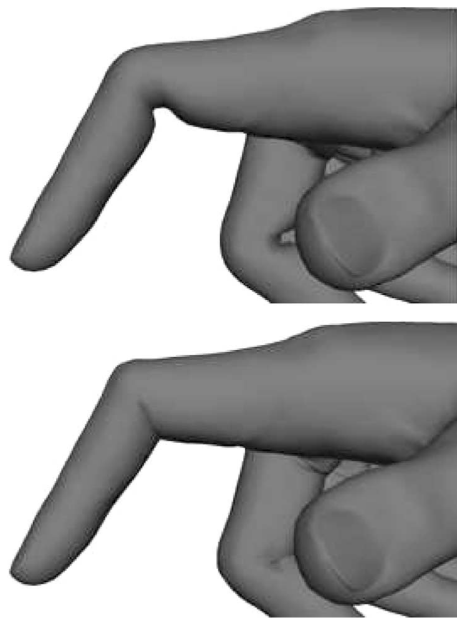

Fig. 4 *Effects of weights on the shape of the model*

Intersecting surfaces (right) are preferred over shrunken skin (left) as they produce a more natural shape

2.4 Hand model as a function

The varying state $\boldsymbol{p}$ of the hand model consists of the translation and rotation of the palm, and the joint angles of the phalanges. This is coded in the form of a function $\mathcal{F}$ that, for given $\boldsymbol{p}$ and intrinsic model parameters $\mathcal{M}$, maps a point $\hat{\boldsymbol{x}}_m$ given in 3-D hand model coordinates to the predicted 3-D position $\hat{\boldsymbol{x}}_c$ in camera coordinates. For simplicity, a pseudo-orthographic projection is assumed:

$$\hat{\boldsymbol{x}}_c = \mathcal{F}(\hat{\boldsymbol{x}}_m, \boldsymbol{p}, \mathcal{M}). \tag{2}$$

3 Stochastic gradient approach

3.1 Observations

Tracking proceeds by matching the model described in the previous Section against dense 3-D measurements extracted at video rate. The input speed of our structured light sensor [Note 3] is thus quite high, but the 3-D output is not produced at this speed. Hence, the 3-D results are provided at a rate that is too low to support on-line tracking. This said, alternative structured light systems exist that do reach the necessary speed [20, 21].

In order not to fall victim to erroneous 3-D data, which tend to cluster near the rim of the hand, we apply skin colour detection to mask the 3-D data.

3.2 Objective function

For a tracked point $\hat{\boldsymbol{x}}_m$ on the hand model, we seek to minimise the distance between its predicted 3-D position $\hat{\boldsymbol{x}}_c$ and observed 3-D position $\boldsymbol{x}_c$. As we have no information about this corresponding, observed point, we rather minimise, in a way similar to some iterative closest point (ICP) [22] implementations, the distance of $\hat{\boldsymbol{x}}_c$ to the tangent plane at the point on the observed 3-D surface with the same image projection, i.e. $\hat{\boldsymbol{x}}'_c = (\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}, \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}))$. This pseudo-orthographic projection is illustrated in Fig. 5. If the closest point on this plane is $\boldsymbol{x}_c$, the tracker minimises the following sum-squared likelihood function over a sample set S_i :

$$E = \sum_{S_i} \frac{1}{2} (\|\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c\|^2), \tag{3}$$

where $\|\cdot\|$ denotes the L_2-norm.

The normal $\boldsymbol{n}$ on the observed 3-D surface at the point $\hat{\boldsymbol{x}}_c$ is obtained as:

$$\boldsymbol{n}^* = \left[\left.\frac{\partial \mathcal{Z}}{\partial x}\right|_{\hat{\boldsymbol{x}}'_c}, \left.\frac{\partial \mathcal{Z}}{\partial y}\right|_{\hat{\boldsymbol{x}}'_c}, -1 \right]^T \quad \Rightarrow \quad \boldsymbol{n} = \frac{\boldsymbol{n}^*}{\|\boldsymbol{n}^*\|}, \tag{4}$$

where ^{T} denotes the matrix transpose.

One can therefore deduce from Fig. 5 that the observed 3-D position $\boldsymbol{x}_c$ can be computed as:

$$\boldsymbol{x}_c = \hat{\boldsymbol{x}}_c + [(\hat{\boldsymbol{x}}'_c - \hat{\boldsymbol{x}}_c) \cdot \boldsymbol{n}]\boldsymbol{n}$$

$$\begin{aligned} \hat{\boldsymbol{x}}_c - \boldsymbol{x}_c &= \left[\left(\begin{pmatrix} \hat{\boldsymbol{x}}_{c,x} \\ \hat{\boldsymbol{x}}_{c,y} \\ \hat{\boldsymbol{x}}_{c,z} \end{pmatrix} - \begin{pmatrix} \hat{\boldsymbol{x}}_{c,x} \\ \hat{\boldsymbol{x}}_{c,y} \\ \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) \end{pmatrix} \right) \cdot \boldsymbol{n} \right] \boldsymbol{n} \\ &= \left[\begin{pmatrix} 0 \\ 0 \\ \hat{\boldsymbol{x}}_{c,z} - \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) \end{pmatrix} \cdot \boldsymbol{n} \right] \boldsymbol{n} \\ &= \frac{[\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}]\boldsymbol{n}^*}{\|\boldsymbol{n}^*\|^2}. \end{aligned} \tag{5}$$

Note 3: http://www.eyetronics.com

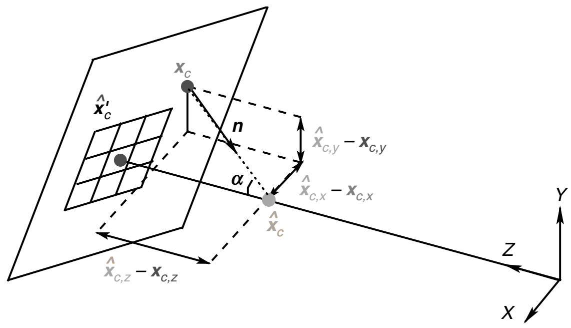

Fig. 5 *Calculation of the error function*

An approximation to the closest point on the surface is calculated by projecting the model point $\hat{\boldsymbol{x}}_c$ onto the tangential plane at $\hat{\boldsymbol{x}}'_c = (\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}, \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}))$. This yields the approximation $\boldsymbol{x}_c$. $\mathcal{Z}$ is the depth map provided by the 3-D sensor in camera coordinates

Then, we derive the components of (3) from (5):

$$\begin{aligned}\hat{\boldsymbol{x}}_{c,x} - \boldsymbol{x}_{c,x} &= \frac{[\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}]\boldsymbol{n}^*_x}{\|\boldsymbol{n}^*\|^2} \\ &= \frac{\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}}{\|\boldsymbol{n}^*\|^2}\frac{\partial \mathcal{Z}}{\partial x}\end{aligned} \tag{6}$$

$$\begin{aligned}\hat{\boldsymbol{x}}_{c,y} - \boldsymbol{x}_{c,y} &= \frac{[\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}]\boldsymbol{n}^*_y}{\|\boldsymbol{n}^*\|^2} \\ &= \frac{\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}}{\|\boldsymbol{n}^*\|^2}\frac{\partial \mathcal{Z}}{\partial y}\end{aligned} \tag{7}$$

$$\begin{aligned}\hat{\boldsymbol{x}}_{c,z} - \boldsymbol{x}_{c,z} &= \frac{[\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}]\boldsymbol{n}^*_z}{\|\boldsymbol{n}^*\|^2} \\ &= \frac{\hat{\boldsymbol{x}}_{c,z} - \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y})}{\|\boldsymbol{n}^*\|^2}\end{aligned} \tag{8}$$

In order to cope with noisy data, we repair small holes in the depth map by interpolation. If no depth at all (i.e. background) is found at $(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y})$, we set $\boldsymbol{x}_c$ to the nearest location in $\mathcal{Z}$ with depth, and do not try to match depth ($\boldsymbol{x}_{c,z} := \hat{\boldsymbol{x}}_{c,z}$).

3.3 Computation of the gradient

To derive the gradient of E in the state space of poses $\boldsymbol{p}$, let $\boldsymbol{J}_{\mathcal{K}}$ denote the Jacobian of a function $\mathcal{K}: \mathbb{R}^n \rightarrow \mathbb{R}^m$, i.e. the $m \times n$ matrix of partial derivatives of the m outputs of $\mathcal{K}$ with respect to its n inputs. We calculate the gradient $\boldsymbol{g}_i$ of our objective E at iteration i by backwards chaining of derivatives:

$$\boldsymbol{g}_i \equiv \frac{\partial E}{\partial \boldsymbol{p}_i} = \boldsymbol{J}^T_{E \circ \mathcal{F}} = \boldsymbol{J}^T_{\mathcal{F}} \boldsymbol{J}^T_E \tag{9}$$

where ^{T} denotes the matrix transpose, and $\circ$ function composition. Note that $\boldsymbol{J}^T_E$, and consequently $\boldsymbol{g}_i$, are column vectors since E must be a scalar function.

The first-order stochastic gradient descent approach to find a minimum of E can then be written as follows:

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - a\boldsymbol{g}_i \tag{10}$$

where i denotes the ith iteration and the step size is represented by a.

3.4 Stochastic sub-sampling

The object function E (equation (3)) is evaluated by considering only a limited set of points to speed up the tracking. In this paper, only 45 points were used. In a 26-dimensional search space, 45 points are very few. For this reason, they are spread in the following way: 2 points per phalanx and 15 points on the palm (or the back of the palm according to the visibility). Every 5 iterations, a visibility algorithm is run (depth-buffer algorithm) and if some phalanges are occluded, the sampled points dedicated to them are replaced by randomly chosen points on the visible part of the hand. This is illustrated in Fig. 6.

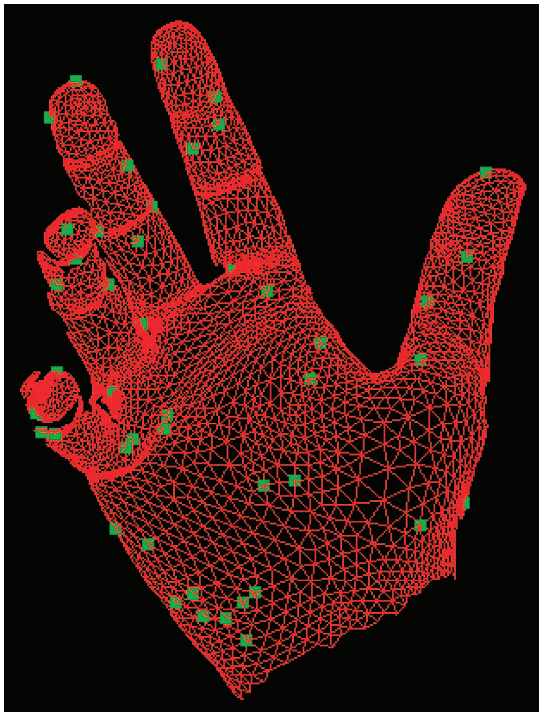

Fig. 6 *Illustration of a random subsampling performed on the hand model with 2 points per phalanx and 15 points for the visible part of the palm*

When a hand part is not visible, its points are replaced by randomly chosen points on any other visible part

The discrete nature of the sampling process and the noise in the 3-D measurement introduce many local minima in the optimisation function where the algorithm could get stuck. By randomly changing the sampling for each iteration step, these spurious minima will also change, allowing the optimisation to proceed towards the true minimum.

4 Rapid stochastic gradient descent

Having set up a gradient-based framework for 3-D visual tracking, we now address the issue of how to implement the gradient descent; how the error or 'loss function' E is efficiently minimised with respect to the hand model parameters $\boldsymbol{p}$.

4.1 Problems with conventional methods

There is an arsenal of well-known techniques for minimising multidimensional functions. Some methods do not require the calculation of a gradient or higher order derivatives. Of these, the Powell method [23] aims to find N conjugate directions and then minimises along them. In our case where the function to minimise is nonlinear, Powell's method needs many iteration steps and is therefore slow.

The APF algorithm, which can handle nonlinear optimisation problems, belongs to the same class. As this method is derived from a particle filter based on simulated annealing, several hypotheses are handled simultaneously, which increases the robustness of the tracking. Furthermore, APF searches for a global minimum without being distracted by local minima but is computationally expensive.

When derivatives are available, nonlinear optimisation problems are often solved by second-order gradient techniques such as the Levenberg-Marquardt algorithm [24, 25] or truncated quasi-Newton methods like the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [23]. The basic idea of these methods is to build up, iteratively, an approximation to the inverse of the Hessian, which is then used to update the model parameters in large steps. This makes it hard to enforce constraints on the parameters: the farther a single step may take us outside the feasible region, the more difficult it becomes to return to it. Interior point methods use barrier functions to confine the search to the feasible region, but they require solving a series of optimisation problems while annealing a Lagrange multiplier, making them too expensive for our purposes.

Conjugate gradient (CG) techniques [23] are computationally cheaper needing only linear cost per iteration. They first minimise along the direction of the gradient and then construct a new direction that is conjugate to all previous directions. In contrast to Powell's method, they employ the gradient for finding these directions. CG methods have the drawback that each iteration strongly depends on the preceding ones. Thus CG must be restarted whenever it strays from the feasible region, at a large loss in performance and high computational expense. Furthermore, we observed that CG does not tolerate well the noise inherent in our approximation of the gradient by sampling and that it converges poorly on highly nonlinear problems.

Simple first-order gradient descent (GD), by contrast, handles noisy gradients well, and since it makes small, incremental steps, mapping parameter values back into the feasible region is straightforward. Its main drawback is slow convergence in ill-conditioned systems, whose energy landscape is characterised by long, narrow valleys (see Fig. 7). Often this is due to the gradient with respect to different subsets of parameters having widely differing magnitude; for instance, this is observed for translation against rotation parameters of our hand model. We have therefore developed the stochastic meta-descent (SMD) technique for local step size adaptation; it is described in detail below.

4.2 Local gradient step size adaptation

Convergence can be accelerated considerably by scaling the gradient for each parameter by its own step size. In other words, the parameter vector $\boldsymbol{p}$ is updated via

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - \boldsymbol{a}_i \cdot \boldsymbol{g}_i; \quad \boldsymbol{g}_i \equiv \sum_{S_i} \boldsymbol{J}_{\mathcal{F}}^T \boldsymbol{J}_E^T \tag{11}$$

where $\cdot$ denotes the Hadamard (i.e. component-wise) product. The vector $\boldsymbol{a}$ of local step sizes is in effect a diagonal conditioner for the gradient system. By observing the sign of the autocorrelation of the gradient, one obtains simple strategies for adapting $\boldsymbol{a}$ during optimisation [26]: the step size is decreased along directions where the gradient oscillates (i.e. $\boldsymbol{g}_i \cdot \boldsymbol{g}_{i-1} < 0$), and increased along those where it does not.

This approach should not be used in combination with stochastic gradient estimates, however, since the nonlinear sign function does not commute with the expectation operator [27]. A more principled alternative that does admit stochastic approximation of gradients can be derived from the notion of adapting $\boldsymbol{a}$ by a meta-level gradient descent in E as well [26]. To allow $\boldsymbol{a}$ to cover a wide dynamic range while remaining strictly positive, the meta-level descent is best performed on $\ln \boldsymbol{a}$ [28]. Assuming that each element of $\boldsymbol{a}$ affects the objective E only through the corresponding element of $\boldsymbol{p}$, such that the chain rule can be used, we obtain

$$\begin{aligned} \ln \boldsymbol{a}_i &= \ln \boldsymbol{a}_{i-1} - \mu \frac{\partial E}{\partial \boldsymbol{p}_i} \cdot \frac{\partial \boldsymbol{p}_i}{\partial \ln \boldsymbol{a}_{i-1}} \\ &= \ln \boldsymbol{a}_{i-1} + \mu \boldsymbol{g}_i \cdot \boldsymbol{v}_i, \end{aligned} \tag{12}$$

where μ is a scalar meta-step size, and $\boldsymbol{v}_i$ characterises the dependence of parameters on their step sizes. From (11) we find that

$$\boldsymbol{v}_{i+1} \equiv -\frac{\partial \boldsymbol{p}_{i+1}}{\partial \ln \boldsymbol{a}_i} = \boldsymbol{a}_i \cdot \boldsymbol{g}_i \tag{13}$$

which, inserted into (12), gives us the familiar autocorrelation $\boldsymbol{g}_i \cdot \boldsymbol{g}_{i-1}$ of the gradient, now without the problematic sign function [27, 29]. Finally, exponentiating (12) gives

$$\begin{aligned} \boldsymbol{a}_i &= \boldsymbol{a}_{i-1} \cdot \exp(\mu \boldsymbol{g}_i \cdot \boldsymbol{v}_i) \\ &\approx \boldsymbol{a}_{i-1} \cdot \max\left(\frac{1}{2}, 1 + \mu \boldsymbol{v}_i \cdot \boldsymbol{g}_i\right). \end{aligned} \tag{14}$$

The linearization $e^u \approx \max((1)/(2), 1+u)$ eliminates the expensive exponentiation for each weight update, while ensuring that the multiplier for $\boldsymbol{a}$ remains positive. In any case, since μ must be chosen sufficiently small for the meta-level descent in $\ln \boldsymbol{a}$ to be stable, this bilinear approximation is sufficiently accurate.

4.3 Stochastic meta-descent (SMD)

The simple definition (13) for $\boldsymbol{v}$ has the severe disadvantage of failing to take into account long-term dependencies of parameter values $\boldsymbol{p}$ on step sizes $\boldsymbol{a}$ (see Fig. 8). Ideally, one would like to model the effect of the current step size on future weights (Fig. 8*a*). In recognition of this problem, the term $\boldsymbol{g}_{i-1}$ in the gradient's autocorrelation is sometimes replaced with an exponential running average of past gradients [26]. Although such *ad hoc* smoothing does improve performance, it does not model long-term dependencies, the average being one of immediate, single-step effects.

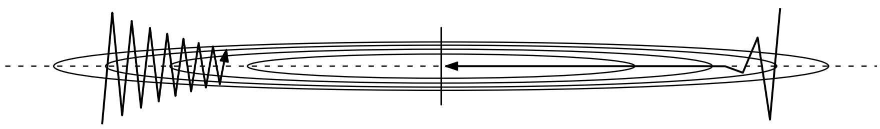

Fig. 7 *Gradient descent*

Gradient descent with a single, global step size (left, 15 steps) traverses narrow valleys in the objective function (contour lines) very inefficiently. For axis-aligned valleys, local step size adaptation (right, 15 steps) greatly accelerates convergence by decreasing the step size along directions where the gradient oscillates, and increasing it along those where it does not. The effect is that of a diagonal conditioner that is continually adapted to the local shape of the objective function

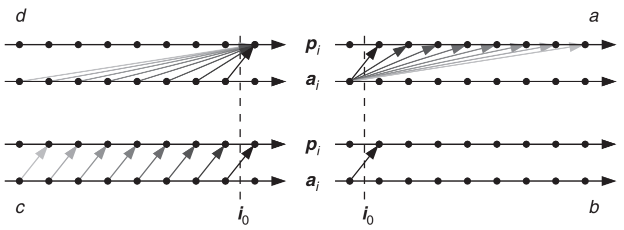

Fig. 8 *Dependence of a parameter p on its step size a, considered at the iteration i_0*

Shown clockwise from upper right
a Future parameters depend on the current step size; the dependence diminishes over time owing to the ongoing adaptation of *a*
b Standard step size adaptation methods capture only the immediate effect
c Even when past gradients are exponentially smoothed
d SMD, by contrast, iteratively models the dependence of the current parameter on an exponentially weighted past history of step sizes, and is able to capture long-range effects missed by other algorithms

By contrast, [30] models the long-term effect of $\boldsymbol{a}$ on future parameter values in a linear system by carrying the relevant partials forward through time. This results in an iterative update rule for $\boldsymbol{v}$ extended to nonlinear systems. The resulting SMD algorithm redefines $\boldsymbol{v}$ as an exponential average of the effect of all past step sizes on the new parameter values:

$$\boldsymbol{v}_{i+1} \equiv -\sum_{k=0}^{\infty} \lambda^k \frac{\partial \boldsymbol{p}_{i+1}}{\partial \ln \boldsymbol{a}_{i-k}}. \tag{15}$$

The factor $0 \leq \lambda \leq 1$ governs the time scale over which long-term dependencies are taken into account. Inserting (11) into (15) gives

$$\begin{aligned} \boldsymbol{v}_{i+1} &= -\sum_{k=0}^{\infty} \lambda^k \frac{\partial \boldsymbol{p}_i}{\partial \ln \boldsymbol{a}_{i-k}} + \sum_{k=0}^{\infty} \lambda^k \frac{\partial(\boldsymbol{a}_i \cdot \boldsymbol{g}_i)}{\partial \ln \boldsymbol{a}_{i-k}} \\ &\approx \lambda \boldsymbol{v}_i + \boldsymbol{a}_i \cdot \boldsymbol{g}_i + \boldsymbol{a}_i \cdot \left[\frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{p}_i^T} \sum_{k=0}^{\infty} \lambda^k \frac{\partial \boldsymbol{p}_i}{\partial \ln \boldsymbol{a}_{i-k}} \right] \\ &= \lambda \boldsymbol{v}_i + \boldsymbol{a}_i \cdot (\boldsymbol{g}_i - \lambda \boldsymbol{H}_i \boldsymbol{v}_i) \end{aligned} \tag{16}$$

where $\boldsymbol{H}_i$ denotes the instantaneous Hessian at iteration i. This iterative update is in fact an efficient stochastic variant of the Levenberg-Marquardt trust region approach. For the sake of simplicity, (16) ignores partials of the components of $\boldsymbol{a}_i$ with respect to their own past values; not doing so would imply meta-meta-level adaptation, complicating matters more than is worth in terms of performance.

Since the Hessian of a system with n parameters has $O(n^2)$ entries, its appearance in (16) might suggest that SMD is a computationally expensive algorithm. Fortunately this is not the case, since there are very efficient indirect methods for computing the product of the Hessian with an arbitrary vector [31]. To prevent negative eigenvalues from causing (16) to diverge, SMD uses an extended Gauss-Newton approximation that also admits a fast matrix-vector product. In our case, this is given by

$$\boldsymbol{H}_i \boldsymbol{v}_i \approx \sum_{S_i} \boldsymbol{J}_{\mathcal{F}}^T \boldsymbol{H}_E \boldsymbol{J}_{\mathcal{F}} \boldsymbol{v}_i \tag{17}$$

with the multiplication by $\boldsymbol{J}_{\mathcal{F}}$ and its transpose performed by algorithmic differentiation.

4.4 Incorporation of constraints

To assist the search in a high-dimensional space, the use of constraints is essential. The classical approach to solve constrained optimisation problems is the method of Lagrange multipliers [32] but as explained earlier in Section 4.1, they are computationally too expensive for our purpose.

In the case of human body tracking, different solutions have been proposed so as to enforce the anatomical joint limits [4, 33]. Wachter *et al.* [33] track a 10–15 DOFs human body using an iterated extended Kalman filter. To enforce the anatomical joint limits (hard constraints in this case), the constraints are enforced after the prediction step, although they may be violated during the update step. For the same task, Sminchisescu *et al.* [4] estimate the body parameters by using a second order trust region method. Hard constraints are taken into account by projecting the gradient onto the current active constraint set. Wu *et al.* [2] learned the anatomical constraints of the hand via a cyberglove. They reduced the state space to a 7-dimensional subspace approximated by the union of basis configurations. Then, the parameter estimation is performed by importance sampling in the learned subspace.

In robotics, the problem has been stated as follows: $\boldsymbol{p}$ is the state of an N-DOFs kinematic chain and $\boldsymbol{x}$ is a M-dimensional vector indicating the position and orientation of the manipulator's end effector (in our case, a finger tip). Then, one wants to minimize the kinematic function:

$$\Delta \boldsymbol{x} = J \Delta \boldsymbol{p} \tag{18}$$

In order to solve (18), one has to compute the inverse of the Jacobian (if it exists). However, in redundant cases (N > M), there are an infinity of solutions. The classical way to overcome this problem is to compute the pseudoinverse of the Jacobian $J^+ = J^T(JJ^T)^{-1}$ [34]. Equation (18) can now be expressed as:

$$\Delta \boldsymbol{p} = J^+ \Delta \boldsymbol{x} + (I - J^+ J) z \tag{19}$$

where z is an arbitrary vector expressed in the $\Delta \boldsymbol{p}$ space. $(I - J^+ J)$ is the projection operator onto the null space of J. Girard *et al.* [34] use z to constraint joint angles while animating legged figures. However, we choose a simpler approach that can be incorporated elegantly in the SMD framework.

We enforce the constraints by means of a function that after each update (11) maps the parameters back into the feasible region:

$$\boldsymbol{p}_{i+1}^c = \text{constrain}(\boldsymbol{p}_{i+1}). \tag{20}$$

Since SMD uses the gradient not only to update the parameters $\boldsymbol{p}$, but also to adjust $\boldsymbol{a}$ and $\boldsymbol{v}$, we must make it aware of the constraints on $\boldsymbol{p}$. We do this by calculating a hypothetical 'constrained' gradient $\boldsymbol{g}^c$ which, when applied in an unconstrained setting, would cause the same parameter change that we would observe with constraints. In other words, we require that

$$\boldsymbol{p}_{i+1}^c = \boldsymbol{p}_i^c - \boldsymbol{a}_i \cdot \boldsymbol{g}_i^c \quad \Rightarrow \quad \boldsymbol{g}_i^c = \frac{\boldsymbol{p}_i^c - \boldsymbol{p}_{i+1}^c}{\boldsymbol{a}_i}. \tag{21}$$

Using this constrained gradient instead of the ordinary one in (16) enables the SMD step size adaptation machinery to function well in our constrained setting.

4.5 Summary of gradient descent iteration

To summarise, each iteration of our gradient-based SMD optimisation algorithm comprises, in the following order, of:

(1) calculate the gradient (9),
(2) update gradient step sizes (14),
(3) update hand model parameters (11),
(4) apply hand model constraints (20),
(5) calculate constrained gradient (21),
(6) calculate the $\boldsymbol{Hv}$ product (17),
(7) update SMD $\boldsymbol{v}$ vector (16).

5 Results

The experiments were processed on a Sunfire 1 GHz.

5.1 Setting SMD parameters

To illustrate the SMD tracker we show results for several 3-D hand tracking sequences. The 3-D data are acquired at 12.5 frames per second and 720×576 pixels. Although SMD is very effective at adapting local learning rates to changing requirements, it is nevertheless sensitive to their initial values.

An intuitive way to initialise the step sizes is to observe the value of the gradient in the first few iterations, to estimate the order of magnitude for each dimension. To evaluate both influences of μ and λ on the speed of our tracker, we ran the SMD tracker several times with different

Table 1: Comparison of the computation times for the experiment presented in Fig. 9

Method	SMD	BFGS	GD	Powell	APF
Time (s/frame)	4.7	8.4	13.5	127.3	131.0

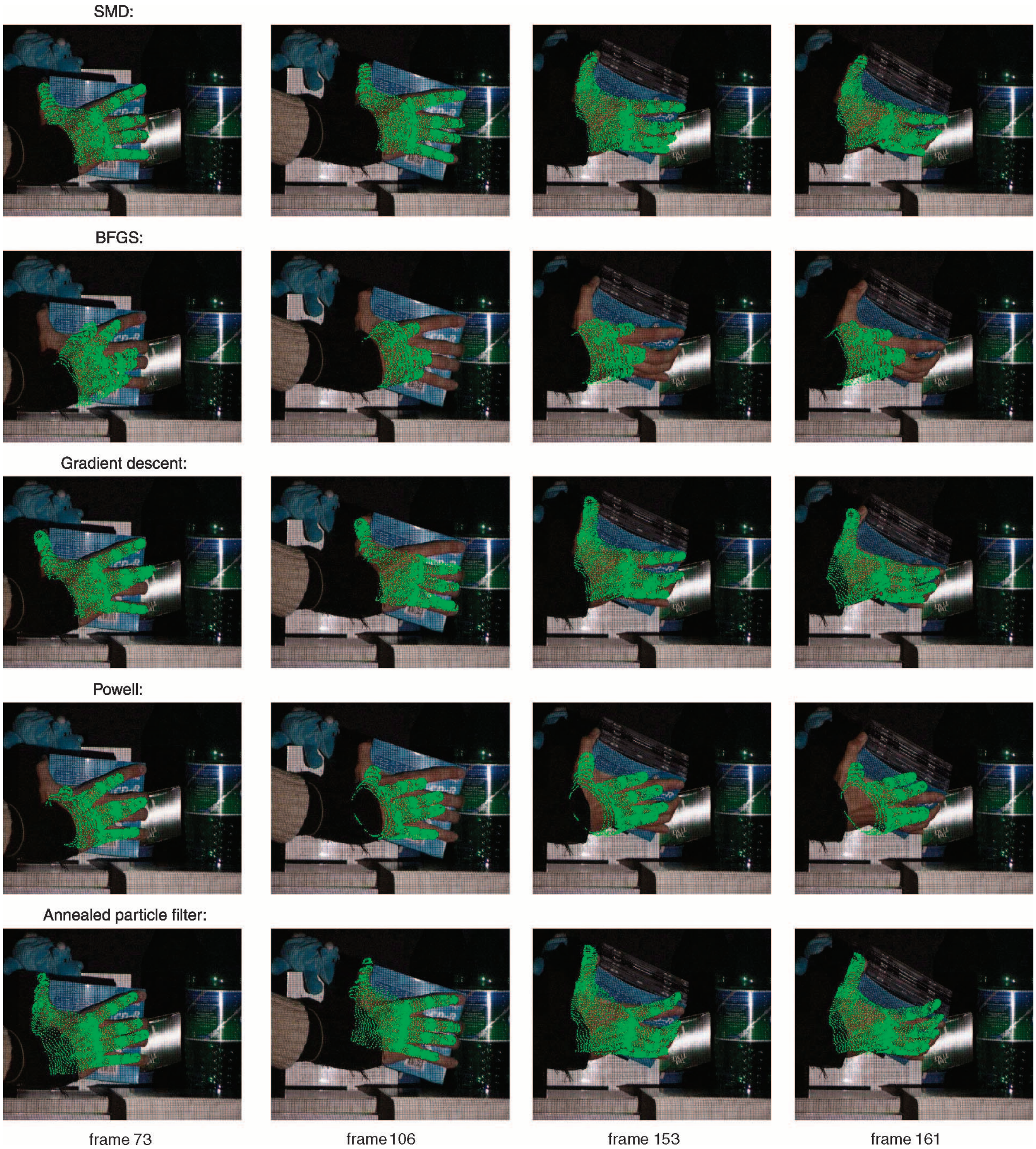

Fig. 9 *Comparison between SMD and several alternative optimisation algorithms*
From top to bottom the results of the SMD, BFGS, GD, Powell and APF algorithms are shown. The model is superimposed on the images. See the text for a discussion

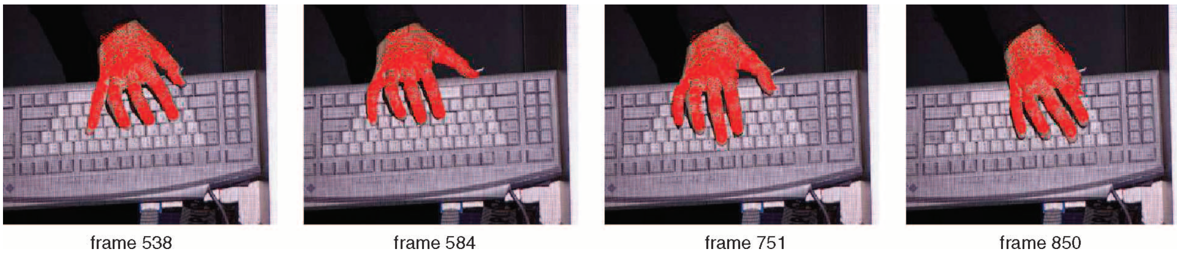

Fig. 10 *Tracking results for a keyboard typing sequence*

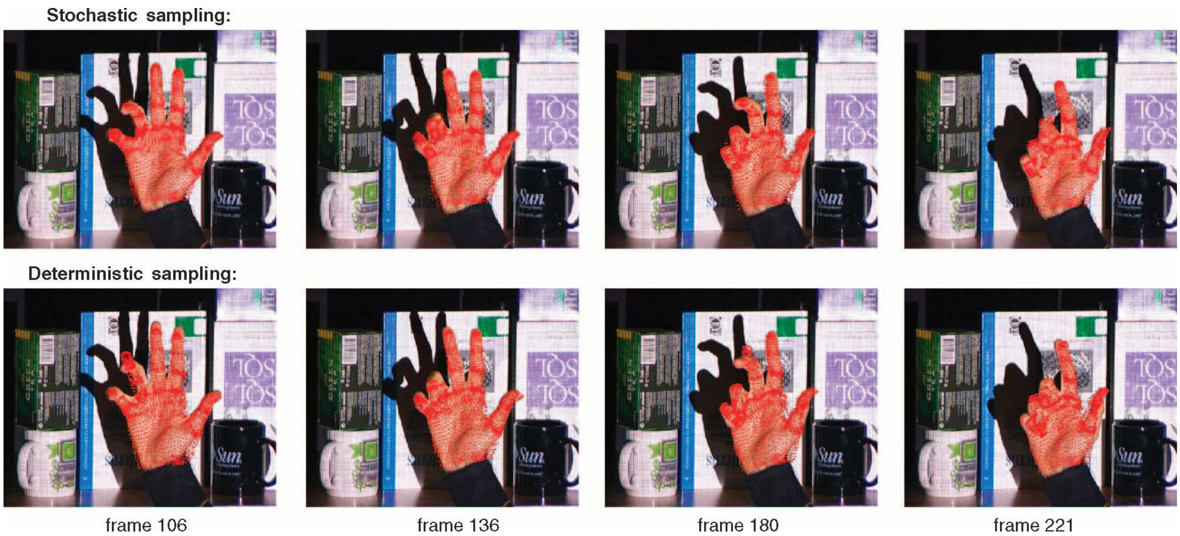

Fig. 11 *Tracking during self-occlusion: stochastic sampling (top row) against deterministic sampling (bottom row)*

values of μ and λ. Processing time for μ around 0.09 and λ close to 1 came out to be lowest on average. Therefore, we use these settings in the subsequent experiments. More in general, SMD was efficient for μ between 0.08 and 0.1 and λ between 0.9 and 1 when the step sizes are tuned towards long strides. μ is scalable depending on the application, i.e. the gradient of the cost function. Furthermore, depending on the application, these optimal intervals should be reconsidered.

5.2 Efficiency of SMD against other methods

First of all, we compare our approach to some major, alternative algorithms (see Section 4.1) for multidimensional minimisation. The results are presented for a grasping sequence in front of a cluttered background. The hand is rotated and some fingers are occluded. The video consists of 165 frames. The results of our SMD approach are shown in the top row of Fig. 9. Subsequent rows present the results of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [23], the gradient descent (GD) [23], the Powell [23] and the annealed particle filter (APF) [3] algorithms. For BFGS and Powell we used the implementations of the VXL package [Note 4]. GD was applied by using SMD and setting μ and λ to zero. The APF version tried here was our own implementation, which is not yet optimised for speed (the same holds for the SMD method however).

For all experiments 45 points were sampled on the hand model except for the APF algorithm, which required 200 points, 300 samples and 12 layers. Indeed, APF with 45 points failed. In this experiment SMD performed best, not only in terms of accuracy but also computation time (see Table 1). The BFGS algorithm is quite fast too, but gets stuck in a local minimum from which it does not recover. This is partly due to the fact that this method cannot handle noise well and is designed mainly for unconstrained optimisation. GD behaves much better. It also gets stuck in a local minimum, but as it can handle noise better, the false minimum usually affects only a part of the hand such as one digit and not the overall position. Powell's method is rather slow as it does not use gradient information and it loses the correct solution fast. Besides the SMD algorithm, APF provides the most convincing results, which is explained by its search for the global minimum. It can recover from a bad match, but, as Fig. 9 shows, the palm, the thumb and little finger are not always tracked very accurately. Looking at the alternative methods, there is a trade-off between quality and speed (Table 1).

In the next experiment (see Fig. 10), we present results for typing on a keyboard. The speed of the SMD tracker depends on the complexity of the scene but was on average 1.9 seconds per frame for this sequence. This is faster than for the first experiments, owing to the simpler hand poses (more frontal to the 3-D acquisition system and fewer self-occlusions). The largest motions occur in the bending of the fingers, which are handled well by SMD.

5.3 Stochastic against deterministic

Figure 11 illustrates the effect of deterministically sampled points (bottom row) in comparison to stochastic sampling (top row). This sequence, consisting of 230

Note 4: http://vxl.sourceforge.net

frames with a cluttered background, shows a situation where parts of the fingers are not observable as they are hidden by self-occlusion. For the deterministic sampling, the middle and little finger get stuck in a local minimum (late bending of the model), while the stochasticity increases the tracker's chances to stay out of such spurious local minima. In Fig. 12, an extracted depth map from this sequence is shown where the bended fingers are erroneously glued to the palm. Such erroneous information can only be handled through the incorporation of constraints. The processing of the hand tracking took 2.07 seconds per frame on average.

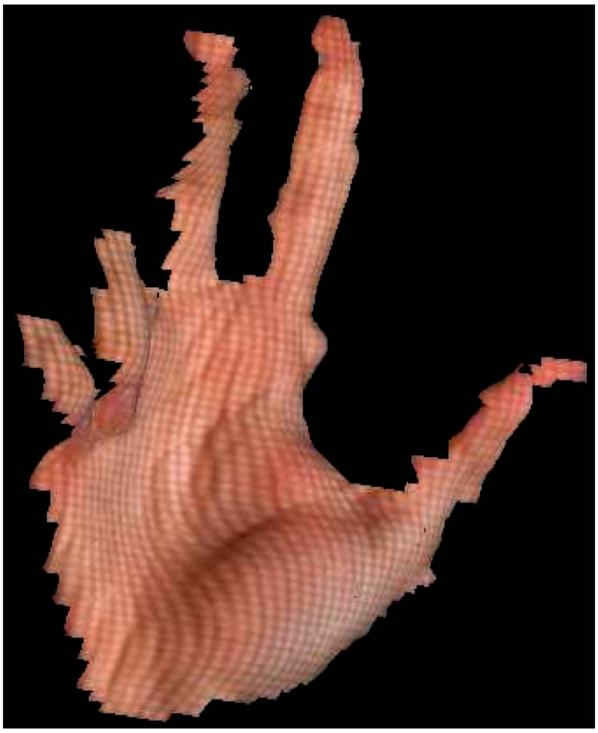

Fig. 12 *Erroneous depth map from the sequence presented in Fig. 11 where the bended fingers are stuck to the palm*

Figure 13 demonstrates the fast convergence of SMD and the advantages provided by a stochastic sampling. It presents the likelihood function (3) projected on the plane with as its two axes the rotation of the palm about the Z axis and its translation along X. The tracker is initialised relatively far from the correct solution as shown in Fig. 13*a* where local minima are around. Owing to the stochastic sampling, the SMD tracker can overcome these local minima as the landscape is continuously changing - except the global minimum - and it reaches the correct solution in 7 iterations. However, Fig. 14 shows the result for a deterministic sampling approach (*a*) and gradient descent (*b*) for the same sequence: with the deterministic sampling, the tracker fails immediately after two iterations. One could argue that using more sampling points in this case would increase the chance of reaching the global minimum, but this would also be computationally more expensive. The gradient descent, with constant step sizes scaled to an optimal value for each dimension and using a 45 points subsampling, needs 23 iterations to converge, 3 times more than the SMD tracker.

5.4 Animation

Our motion capture results are rather noisy owing to imperfect depth maps, sparse observations (subsampling) and limited acquisition rate (12.5 frames per second)

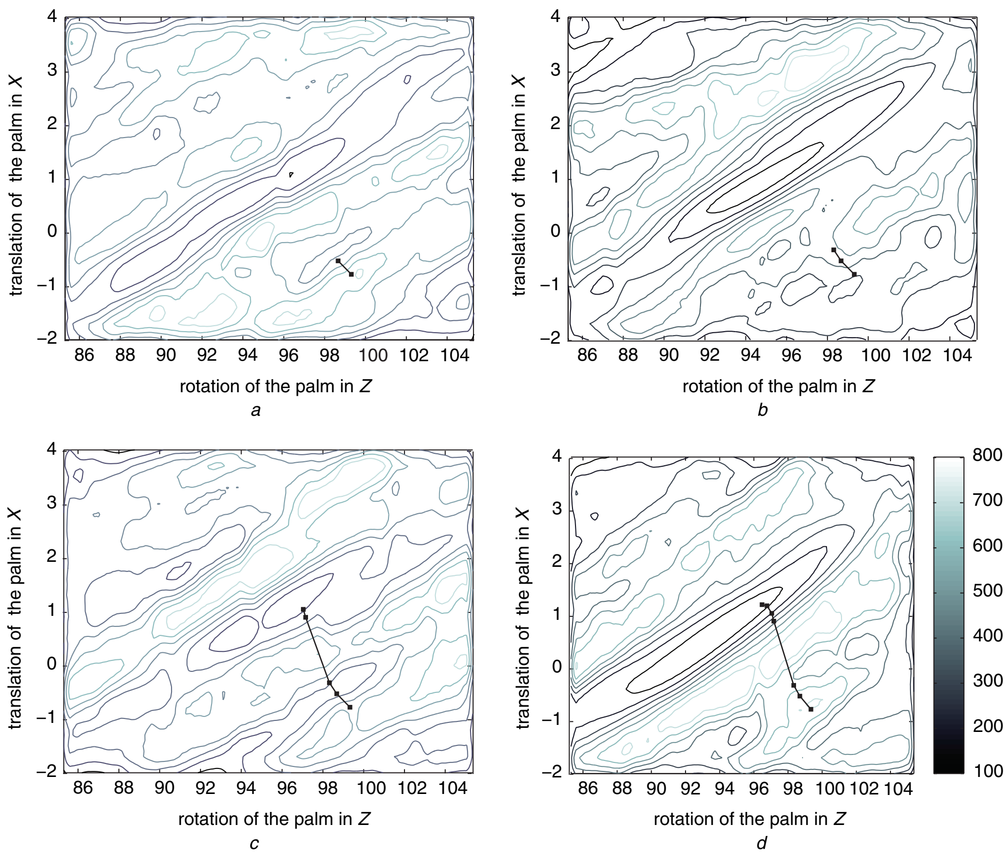

Fig. 13 *Evolution of the SMD algorithm while tracking (i.e. with stochastic sampling). This evolution is projected on the translation in X and rotation in Z of the palm. The convergence is reached in 7 iterations*

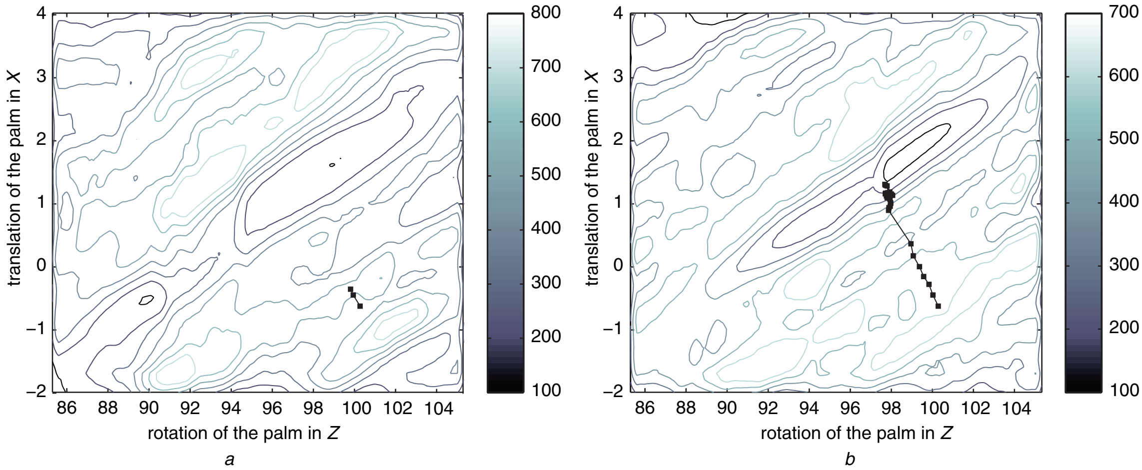

Fig. 14 *Deterministic sampling and gradient descent*

a Evolution of deterministic sampling
b Gradient descent algorithm
This evolution is projected on the translation in *X* and rotation around *Z* of the palm. In the deterministic sampling case, the tracker fails after 2 iterations and in the case of the gradient descent, the optimum is reached after 23 iterations, i.e. more than 3 times the number of iterations required by SMD

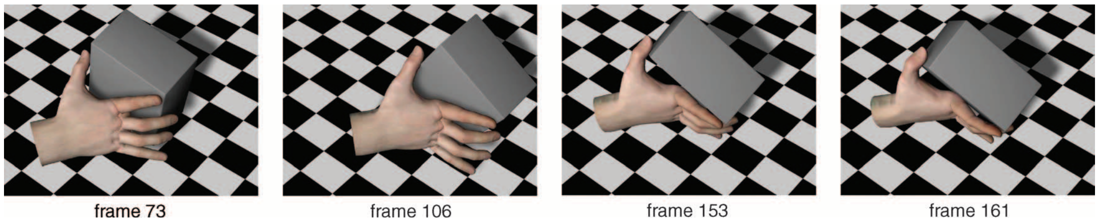

Fig. 15 *Virtual replay: reconstructed sequence of Fig. 9, but from a more overhead viewpoint*

Fig. 16 *Virtual replay: reconstructed sequence of Fig. 10*

The top row represents a frontal view and the bottom row a top view

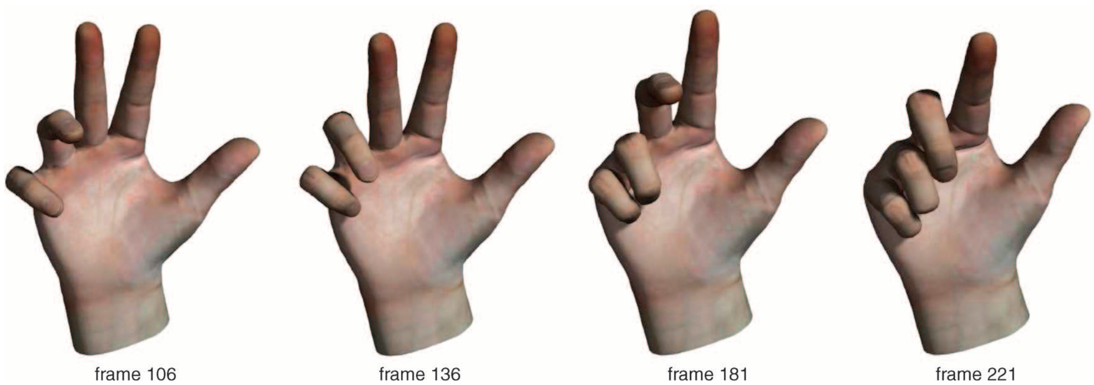

Fig. 17 *Virtual replay: reconstructed sequence of Fig. 11*

Therefore, the tracking results need to be cleaned up in order to use them for animation purposes. There is an extensive literature about motion data filtering [35–37]. We chose to use a simple linear adaptive filter that substitutes each data value by a linear combination of itself and its neighbours [35]. With this simple approach, we obtain satisfying results as shown in Figs. 15–17, which are the virtual replays of Figs. 9–11.

6 Summary and conclusions

Model-based tracking of hand articulations is a challenging task owing to the large number of DOFs. We have proposed a novel SMD tracker, which is based on a rapid stochastic gradient descent approach with adaptive step sizes. Hand constraints can be naturally incorporated into the proposed optimisation process. Our experiments show that the SMD tracker performs robustly and efficiently on rather complex sequences.

While our approach can overcome local minima, it is not guaranteed that the global minimum will be found. The first results of combining several SMD trackers as smart particles within a particle filter framework have been presented in [38], in order to achieve such robustness. Using multiple cameras is another interesting extension and could be implemented easily by modifying the error function. Furthermore, adding pseudo-joints by using the method of Mohr and Gleicher [18] would enhance our model, which could also improve the tracking.

7 Acknowledgments

The authors gratefully acknowledge support for our tracking research by the Swiss SNF NCCR project IM2 and the EU NoE PASCAL.

8 References

1 Sidenbladh, H., Black, M.J., and Fleet, D.J.: 'Stochastic tracking of 3D human figures using 2D image motion'. European Conf. on Computer Vision, 2000, pp. 702–718
2 Wu, Y., Lin, J.Y., and Huang, T.S.: 'Capturing natural hand articulation'. Int. Conf. on Computer Vision, 2001, pp. 426–432
3 Deutscher, J., Blake, A., and Reid, I.: 'Articulated body motion capture by annealed particle filtering'. Int. Conf. on Computer Vision and Pattern Recognition, 2000, pp. 126–133
4 Sminchisescu, C., and Triggs, B.: 'Covariance scaled sampling for monocular 3D body tracking'. Int. Conf. on Computer Vision and Pattern Recognition, 2001, pp. 447–454
5 Rehg, J., and Kanade, T.: 'Model-based tracking of self-occluding articulated objects'. Int. Conf. on Computer Vision, 1995, pp. 612–617
6 Isard, M., and Blake, A.: 'Condensation – conditional density propagation for visual tracking', *Int. J. Comput. Vis.*, 1998, **29**, (1), pp. 5–28
7 Rosales, R., Athitsos, V., Sigal, L., and Sclaroff, S.: '3D hand pose reconstruction using specialized mappings'. Int. Conf. on Computer Vision, 2001, **I**, pp. 378–385
8 Athitsos, V., and Sclaroff, S.: 'An appearance-based framework for 3D hand shape classification and camera viewpoint estimation'. Int. Conf. on Automatic Face and Gesture Recognition, 2002, pp. 40–45
9 Tomasi, C., Petrov, S., and Sastry, A.: '3D tracking = classification + interpolation'. Int. Conf. on Computer Vision, 2003, pp. 1441–1448
10 Stenger, B., Thayananthan, A., Torr, P., and Cipolla, R.: 'Filtering using a tree-based estimator'. Int. Conf. on Computer Vision, 2003, pp. 1063–1070
11 Stenger, B., Thayananthan, A., Torr, P., and Cipolla, R.: 'Hand pose estimation using hierarchical detection'. Int. Workshop on Human-Computer Interaction, 2004, pp. 102–112
12 Catmull, E.E.: 'A system for computer generated movies'. ACM Ann. Conf., August 1972, pp. 422–431
13 Magnenat-Thalmann, N., Laperrire, R., and Thalman, D.: 'Joint dependent local deformations for hand animation and object grasping'. Graphics Interface, 1988, pp. 26–33
14 Komatsu, K.: 'Human skin model capable of natural shape variation', *The Vis. Comput.*, 1988, **4**, (3), pp. 265–271
15 McDonald, J., Toro, J., Alkoby, K., Berthiaume, A., Chomwong, P., Christopher, J., Davidson, M., Furst, J., Konie, B., Lancaster, G., Lytinen, S., Roychoudhuri, L., Sedgwick, E., Tomuro, N., and Wolfe, R.: 'An improved articulated model of the human hand'. 8th Int. Conf. in Central Europe on Computer Graphics, Visualization and Interactive Digital Media, 2000, pp. 306–313
16 Lin, J., Wu, Y., and Huang, T.: 'Modeling human hand constraints'. ARL Federated Laboratory 5th Ann. Symp., 2001, pp. 105–110
17 Lewis, J.P., Cordner, M., and Fong, N.: 'Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation'. Ann. Conf. Ser., ACM SIGGRAPH, 2000
18 Mohr, A., and Gleicher, M.: 'Building efficient, accurate character skins from examples'. Ann. Conf. Ser., ACM SIGGRAPH, 2003, pp. 562–568
19 Buchholz, B., Armstrong, T., and Goldstein, S.: 'Anthropometric data for describing the kinematics of the human hand', *Ergon.*, 1992, **35**, (3), pp. 261–273
20 Koninckx, T., Griesser, A., and Van Gool, L.: 'Real-time range scanning of deformable surfaces by adaptively coded structured light'. 3DIM, 2003, pp. 293–300
21 Rusinkiewicz, S., Hall-Holt, O., and Levoy, M.: 'Real-time 3D model acquisition'. Proc. ACM SIGGRAPH, 2002, pp. 438–446
22 Besl, P., and McKay, H.: 'A method for registration of 3D shapes', *IEEE Trans. Pattern. Anal. Mech. Intell.*, 1992, **14**, pp. 239–256
23 Press, W., Flannery, B., Teukolsky, S., and Vetterling, W.: 'Numerical recipes in C' (Cambridge University Press, 1988)
24 Levenberg, K.: 'A method for the solution of certain non-linear problems in least squares', *Q. J. Appl. Math.*, 1944, **II**, (2), pp. 164–168
25 Marquardt, D.: 'An algorithm for least-squares estimation of non-linear parameters', *J. Soc. Ind. Appl. Math.*, 1963, **11**, (2), pp. 431–441
26 Jacobs, R.: 'Increased rates of convergence through learning rate adaptation', *Neural Netw.*, 1988, **1**, pp. 295–307
27 Almeida, L.B., Langlois, T., and Amaral, J.D.: 'Parameter adaptation in stochastic optimization', in Saad, D. (Ed.): 'On-Line Learning in Neural Networks' (ser. Publications of the Newton Institute, Cambridge University Press, 1999), ch. 6, pp. 111–134
28 Kivinen, J., and Warmuth, M.K.: 'Additive versus exponentiated gradient updates for linear prediction'. Proc. 27th Ann. ACM Symp. on Theory of Computing, 1995, pp. 209–218
29 Harmon, M.E., and Baird, III, L.C.: 'Multi-player residual advantage learning with general function approximation'. Wright Laboratory, WL/AACF, Wright-Patterson Air Force Base, OH 45433-7308, Tech. Rep. WL-TR-1065, 1996
30 Sutton, R.S.: 'Gain adaptation beats least squares?'. Proc. 7th Yale Workshop on Adaptive and Learning Systems, 1992, pp. 161–166
31 Pearlmutter, B.A.: 'Fast exact multiplication by the Hessian', *Neural Comput.*, 1994, **6**, (1), pp. 147–160
32 Bertsekas, D.: 'Nonlinear Programming' (Athena Scientific, 2000)
33 Wachter, S., and Nagel, H.: 'Tracking persons in monocular image sequences', *Comput. Vis. Image Underst.*, 1999, **74**, (3), pp. 174–192
34 Girard, M., and Maciejewski, A.: 'Computational modeling for the computer animation of legged figures'. ACM SIGGRAPH, 1985, pp. 263–270
35 Sudarsky, S., and House, D.H.: 'An integrated approach towards the representation, manipulation and reuse of pre-recorded motion'. CA, 2000, pp. 56–61
36 Litwinowicz, P.C.: 'Inkwell: A 2.5d animation system'. SIGGRAPH, 1991, pp. 113–122
37 Bruderlin, A., and Williams, L.: 'Motion signal processing', *Comput. Graph.*, 1995, **29**, (Annual Conference Series), pp. 97–104
38 Bray, M., Koller-Meier, E., and Van Gool, L.: 'Smart particle filtering for 3D hand tracking'. Int. Conf. on Automatic Face and Gesture Recognition, 2004, pp. 675–680