3D HAND TRACKING BY RAPID STOCHASTIC GRADIENT DESCENT USING A SKINNING MODEL

[1]**Matthieu Bray**, [1]**Esther Koller-Meier**, [1]**Pascal Müller**, [1]**Luc Van Gool** and [2]**Nicol N. Schraudolph**

Swiss Federal Institute of Technology (ETH)
Zürich, Switzerland
[1]{bray,ebmeier,pmueller,vangool}@vision.ee.ethz.ch, [2]n@schraudolph.org

Abstract- The main challenge of tracking articulated structures like hands is their large number of degrees of freedom (DOFs). A realistic 3D 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 optimizations 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 reinforce the robustness of our tracker. Experiments show the efficiency of the SMD algorithm in comparison with common optimization methods.

1 INTRODUCTION

Relatively little work has been devoted to the tracking of articulated structures [1, 2, 3, 4, 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 is needed to guarantee a good solution, but this soon becomes infeasible 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 kind of approaches. It proposes a new type of optimization 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 3D data which are acquired with a structured light system. Skin color detection helps to find the hands. Before going into details, we concisely review the literature on the subject.

As said, 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 1GHz 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.

As mentioned, 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. Compared to the previous approach, the annealed particle filter (APF) reduces the number of samples and increases efficiency by a factor of up to 10. Moreover, APF localizes the tracked object even better as it increases the chances of finding the global minimum. A more detailed comparison between SMD and APF is presented in Section 5 which will demonstrate that, to be competitive with SMD in accuracy, APF is slower than our tracker. In order to reach a good accuracy with sparse sampling, Sminchisescu and Triggs [4] have combined global sampling with local optimization based on a gradient descent method. Their method requires 50 seconds per frame to track in a 30-dimensional space (on a 1GHz processor), which is similar to the dimensionality of our case. In comparison, the proposed SMD approach is again faster.

Basically, both Sminchisescu's approach and APF introduce particles with a more sophisticated behavior, which one could coin 'smart particles'. The SMD tracker can also be considered to represent a smart particle, but in this paper the Condensation framework has not been wrapped around it yet. Nevertheless, even a single SMD tracker (or 'particle') can be seen to already track quite effectively. A cost function is minimized with a novel gradient descent method with step sizes that vary over time and between dimensions. In each iteration, the appropriate changes are automatically determined by a meta-level gradient descent. Individual parts of the SMD algorithm have already been published [7] but to the best of our knowledge not the overall method. Furthermore, they have been only applied to neural networks so far. The integration to a visual hand tracking framework needs modifications since due to the noisy data, we do not have access to the gradient everywhere and have to handle these special cases. Moreover, we naturally incorporate constraints which are in general difficult to integrate or computationally expensive. This new technique is more efficient than previously known methods. Additionally, the introduction of some stochasticity into the evaluation of the objective function increases the chance of getting out of local minima.

A similar idea of using gradient descent is taken by Rehg

and Kanade [5]. They describe an articulated object by a set of overlapping templates and minimize the residual between the observed image and the templates by a simple gradient descent. In comparison, we use 3D rather than 2D 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).
The rest of the paper is organized as follows. Section 2 describes the 3D hand model used for the tracking as well as the constraints on its parameters. Section 3 discusses the effect of the stochastic sampling, the objective function which will be optimized and the computation of the gradient. Section 4 describes the SMD optimization. Section 5 shows tracking results on video sequences. Finally, Section 6 concludes the paper.

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 3D scans. Such 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 [8]. In subsequent work, Magnenat-Thalmann *et al*. [9] and Komatsu [10] 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 fast. For the same reason we use polygons instead of NURBS.

2.1 Linear Blend Skinning

In the meantime, such skinning algorithms are included in most commercial 3D modeling and animation software packages and they go by many names. Magnenat-Thalmann *et al*. called it 'Joint-dependent Local Deformation' [9], Lewis *et al*. called it 'Skeleton Subspace Deformation' [11], Alias' Maya calls it 'Smooth Skinning'[1] and in one of the most recent publications in this field, it is called 'Linear Blend Skinning' [12].
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 bending 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, at an arbitrary 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 *i*th joint in configuration c and $M_{i,b}^{-1}$ is the inverse of the global transformation matrix of the ith joint in binding-pose b. Note that $c = b$ results in the same location of $\boldsymbol{v}$ ($\boldsymbol{v}_c = \boldsymbol{v}_b$) as in its binding pose. A more detailed description can be found in [11].

2.2 Authoring the Model

To model the polygonal surface that makes up the hand, we used 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 structure-light system[2]. The final mesh (see Figure 1) consists of 9051 vertices. This resolution gives a good tradeoff for our application: it is accurate enough and the deformation can easily be solved in real-time on a modern personal computer.
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 segments is called 'joint angle'. The human hand is constrained by dependencies [13] which reduce the model to 31 DOFs: 5 joint angles for each finger (3 for flexion, 1 for abduction, 1 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 as shown in Figure 2. McDonald *et al*. [13] give fixed values for these twists, but to the best of our knowledge, the dynamic dependencies between them and the other joints have not yet been fully explored. Therefore we have chosen to consider them as DOFs, moving in a range of $\pm$ 4 degrees. The hard constraints and dependencies of the other angles are the same as used in [14]. The anthropometrical measurements in [13, 15] give very good approximations for joint-positions. In these two papers, statistical relationships between the skin and the underlying skeleton have been established. Figure 1 shows the resulting hand skeleton for our hand model.

After the skeleton is created, its pose has to be adapted to the pose of the polygonal surface or 'skin', which can then be bound to the skeleton. 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 and influences appropriately is not straightforward. Figure 1 illustrates our weights painted on the hand-model. We used a maximum of 3 influences per vertex.
While Linear Blend Skinning is very fast to evaluate and compact in memory and data-structure, it is notorious for

[1] http://www.aliaswavefront.com

[2] http://www.eyetronics.com

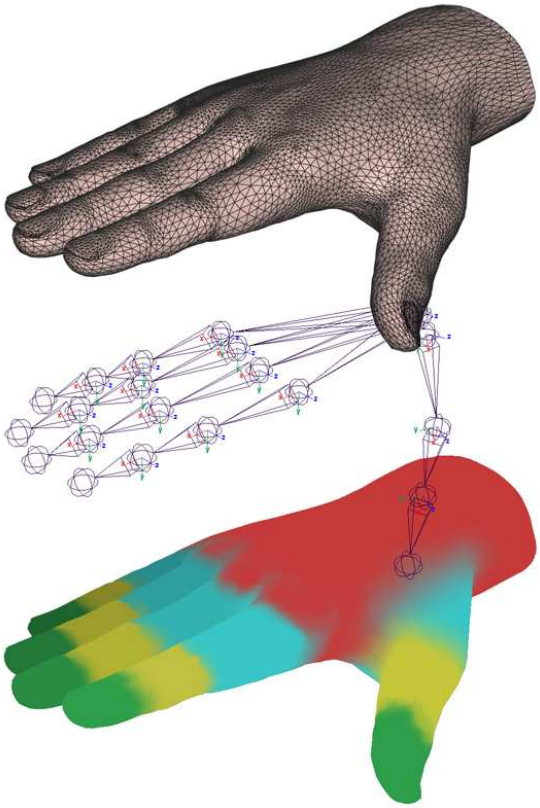


Figure 1: The hand model as polygonal surface (top). The joints in the skeleton (middle) define the locations where the hand should bend. The intensity encoded weights connecting the surface to the skeleton are shown at the bottom. Each intensity represents the influence area of its corresponding joint.

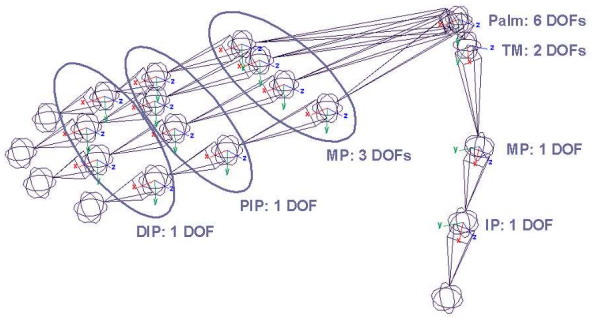


Figure 2: The hand model and its degrees of freedom (DOFs).

its authoring difficulties and its shortcomings. The famous 'shrunken elbow' problem is shown in Figure 3 for a bent finger. Complex deformations like wrinkles etc. can not be represented due to the algorithm's linear nature. This fortunately poses no problems for our tracking task.

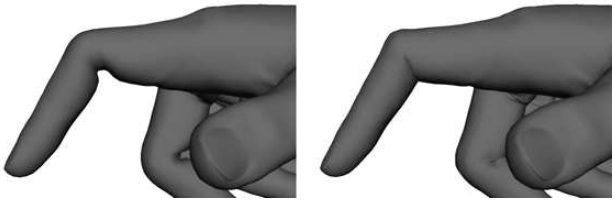


Figure 3: Effects of weights on the shape of the model. Intersecting surfaces (right) are preferred over shrunken skin (left) as they produce a more natural shape.

2.3 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 hand model coordinates into 3D vectors to the predicted 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 Stochastic Sub-Sampling

Tracking proceeds by matching the model of the previous Section against dense 3D measurements extracted at video rate. The input speed of our structured light sensor (from Eyetronics) is thus quite high, but the 3D output is not produced at this speed. Hence, the 3D 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 [16, 17]. The transition to the system described in [16] is in preparation at the time of writing.

In order not to fall victim to erroneous 3D data, which tend to cluster near the rim of the hand, we apply skin color detection to mask the 3D data. To speed up the tracking, the hand is subsampled at 45 stochastically determined points (2 for each visible phalanx and 15 for the palm).

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

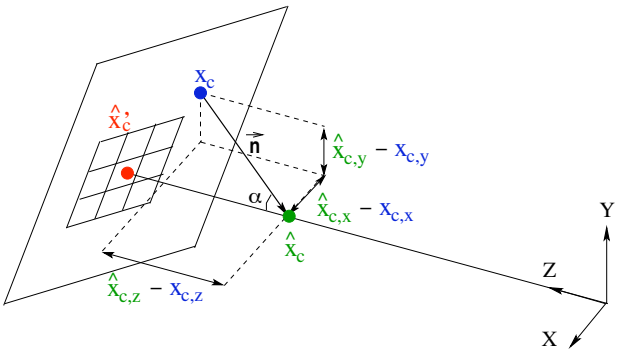


Figure 4: *The calculation of the error function. An approximation to the closest point on the surface is calculated by projecting the model point $\hat{\boldsymbol{x}}_c$ onto the tangential plane at $\hat{\boldsymbol{x}}_c^{'} = (\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}, \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}))$. This yields the approximation $\boldsymbol{x}_c$.*

3.2 Objective Function

For a tracked point $\hat{\boldsymbol{x}}_m$ on the hand model, we seek to minimize the distance between its predicted 3D position $\hat{\boldsymbol{x}}_c$ and observed 3D position $\boldsymbol{x}_c$. As we have no information about this corresponding, observed point, we rather minimize – in a way similar to some ICP implementations – the distance of $\hat{\boldsymbol{x}}_c$ to the tangent plane at the point on the observed 3D surface with the same image projection, i.e. $\hat{\boldsymbol{x}}_c^{'} = (\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}, \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}))$ (still assuming pseudo-orthographic projection) as illustrated in Figure 4

$\mathcal{Z}$ is the depth map provided by the 3D sensor in camera coordinates. If the closest point on this plane is $\boldsymbol{x}_c$, the tracker minimizes the following sum-squared cost function:

$$E(\hat{\boldsymbol{x}}_c) \;=\; \frac{1}{2}\left(||\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c||^2\right), \qquad (3)$$

where $||\cdot||$ denotes the L_2-norm. Our optimization scheme (see Section 4) requires the Jacobian and Hessian of (3), which are:

$$\boldsymbol{J}_E \;=\; [\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c]^T, \qquad \boldsymbol{H}_E = \mathrm{diag}([1,1,1]). \qquad (4)$$

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 ($\boldsymbol{x}_{c,z} := \hat{\boldsymbol{x}}_{c,z}$). In that case the Hessian is set to $\boldsymbol{H}_E = \mathrm{diag}([1,1,0])$.

3.3 Computation of the Gradient

To derive the gradient, let $\boldsymbol{J}_{\mathcal{K}}$ denote the Jacobian of a function $\mathcal{K} : \mathsf{R}^n \rightarrow \mathbb{R}^m$, *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 $\boldsymbol{g}_t$ of our objective E at time t by backwards chaining of derivatives:

$$\boldsymbol{g}_t \;\equiv\; \frac{\partial E}{\partial \boldsymbol{p}_t} \;=\; \boldsymbol{J}_{E\circ\mathcal{F}}^T \;=\; \boldsymbol{J}_{\mathcal{F}}^T \boldsymbol{J}_E^T \qquad (5)$$

where ^{T} denotes the matrix transpose, and $\circ$ function composition. Note that $\boldsymbol{J}_E^T$, and consequently $\boldsymbol{g}_t$, 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:

$$\boldsymbol{p}_{t+1} = \boldsymbol{p}_t - a\,\boldsymbol{g}_t \qquad (6)$$

where t is the time step and the constant step size is represented by a.

4 RAPID STOCHASTIC GRADIENT DESCENT

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

4.1 Problems with Conventional Methods

There is an arsenal of well-known techniques for minimizing multidimensional functions. The simplest methods do not require the calculation of a gradient or higher order derivatives. Of these, the Powell method [18] aims to find N conjugate directions and then minimizes along them. In our case where the function to minimize is non-linear, Powell's method needs many iteration steps and is therefore slow.

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

When derivatives are available, nonlinear optimization problems are often solved by second-order gradient techniques such as the Levenberg-Marquardt algorithm [19, 20] or truncated quasi-Newton method like the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [18]. 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 optimization problems while annealing a Lagrange multiplier, making them too expensive for our purposes.

Conjugate gradient (CG) techniques [18] are computationally cheaper needing only linear cost per iteration. They first minimize along the direction of the gradient and then construct a new direction that is conjugate to all pre-

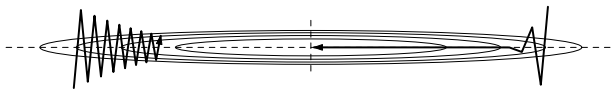


Figure 5: Gradient descent with a single, global step size (left, 15 steps) traverses narrow valleys in the objective function (contour lines) very inefficiently. For axis-aligned valleys, local step size adaptation (right, 15 steps) greatly accelerates convergence by decreasing the step size along directions where the gradient oscillates, and increasing it along those where it doesn't. The effect is that of a diagonal conditioner that is continually adapted to the local shape of the objective function.

vious directions. In contrast to Powells 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 characterized by long, narrow valleys (see Figure 5). 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 *vs*. 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

$$\boldsymbol{p}_{t+1} \;=\; \boldsymbol{p}_t \;-\; \boldsymbol{a}_t \cdot \boldsymbol{g}_t \qquad \boldsymbol{g}_t \;\equiv\; \sum_{S_t} \boldsymbol{J}_{\mathcal{F}}^T \boldsymbol{J}_E^T \qquad (7)$$

where $\cdot$ denotes the Hadamard (*i.e*., component-wise) product, and $\boldsymbol{g}_t$ the gradient of our objective (3) at time t, calculated by backward chaining of derivatives, summed over the points in our stochastic sample S. 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 optimization [21, 22, 23]: the step size is decreased along directions where the gradient oscillates (*i.e*., $\boldsymbol{g}_t \cdot \boldsymbol{g}_{t-1} < 0$), and increased along those where it does not.

This approach should not be used in combination with stochastic gradient estimates however, since the non-linear sign function does not commute with the expectation operator [24]. 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 [21]. To allow $\boldsymbol{a}$ to cover a wide dynamic range while remaining strictly positive, the meta-level descent is best performed on $\ln \boldsymbol{a}$ [25]. 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

$$\begin{aligned} \ln \boldsymbol{a}_t \;&=\; \ln \boldsymbol{a}_{t-1} \;-\; \mu \frac{\partial E}{\partial \boldsymbol{p}_t} \cdot \frac{\partial \boldsymbol{p}_t}{\partial \ln \boldsymbol{a}_{t-1}} \\ &=\; \ln \boldsymbol{a}_{t-1} \;+\; \mu \boldsymbol{g}_t \cdot \boldsymbol{v}_t \,, \end{aligned} \qquad (8)$$

where μ is a scalar meta-step size, and $\boldsymbol{v}_t$ characterizes the dependence of parameters on their step sizes. From (7) we find that

$$\boldsymbol{v}_{t+1} \;\equiv\; -\frac{\partial \boldsymbol{p}_{t+1}}{\partial \ln \boldsymbol{a}_t} \;=\; \boldsymbol{a}_t \cdot \boldsymbol{g}_t \qquad (9)$$

which, inserted into (8), gives us the familiar autocorrelation $\boldsymbol{g}_t \cdot \boldsymbol{g}_{t-1}$ of the gradient, now without the problematic sign function [31, 24]. Finally, exponentiating (8) gives

$$\boldsymbol{a}_t \;=\; \boldsymbol{a}_{t-1} \cdot \exp(\mu\, \boldsymbol{g}_t \cdot \boldsymbol{v}_t) \;\approx\; \boldsymbol{a}_{t-1} \cdot \max(\tfrac{1}{2}\,,\, 1 + \mu\, \boldsymbol{v}_t \cdot \boldsymbol{g}_t) \qquad (10)$$

The linearization $e^u \approx \max(\frac{1}{2}, 1+u)$ 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 (9) for $\boldsymbol{v}$ has the severe disadvantage of failing to take into account long-term dependencies of parameter values $\boldsymbol{p}$ on step sizes $\boldsymbol{a}$ (Figure 6, center/right). In recognition of this problem, the term $\boldsymbol{g}_{t-1}$ in the gradient's autocorrelation is sometimes replaced with an exponential running average of past gradients [21]. Although such ad-hoc smoothing does improve performance, it does not model long-term dependencies, the average being one of immediate, single-step effects.

By contrast, [26] 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 non-linear systems. The resulting *stochastic meta-descent* (SMD) algorithm redefines $\boldsymbol{v}$ as an exponential average of the effect of all past step sizes on the new parameter values:

$$\boldsymbol{v}_{t+1} \;\equiv\; -\sum_{i=0}^{\infty} \lambda^i \frac{\partial \boldsymbol{p}_{t+1}}{\partial \ln \boldsymbol{a}_{t-i}} \,. \qquad (11)$$

The factor $0 \le \lambda \le 1$ governs the time scale over which long-term dependencies are taken into account. Inserting

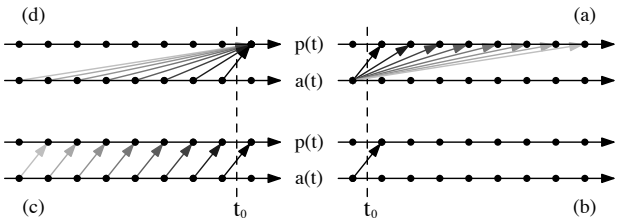


Figure 6: Dependence of a parameter p on its step size a, considered at time t_0. Shown clockwise from upper right: (a) future parameters depend on the current step size; the dependence diminishes over time due to the ongoing adaptation of a. (b) standard step size adaptation methods capture only the immediate effect, even (c) 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.

(7) into (11) gives

$$\begin{aligned} \boldsymbol{v}_{t+1} &= -\sum_{i=0}^{\infty} \lambda^i \frac{\partial \boldsymbol{p}_t}{\partial \ln \boldsymbol{a}_{t-i}} + \sum_{i=0}^{\infty} \lambda^i \frac{\partial (\boldsymbol{a}_t \cdot \boldsymbol{g}_t)}{\partial \ln \boldsymbol{a}_{t-i}} \\ &\approx \lambda \boldsymbol{v}_t + \boldsymbol{a}_t \cdot \boldsymbol{g}_t + \boldsymbol{a}_t \cdot \left[\frac{\partial \boldsymbol{g}_t}{\partial \boldsymbol{p}_t^T} \sum_{i=0}^{\infty} \lambda^i \frac{\partial \boldsymbol{p}_t}{\partial \ln \boldsymbol{a}_{t-i}} \right] \\ &= \lambda \boldsymbol{v}_t + \boldsymbol{a}_t \cdot (\boldsymbol{g}_t - \lambda \boldsymbol{H}_t \boldsymbol{v}_t) \end{aligned} \quad (12)$$

where $\boldsymbol{H}_t$ denotes the instantaneous Hessian at iteration t. This iterative update is in fact an efficient stochastic variant of the Levenberg-Marquardt trust region approach. For the sake of simplicity, (12) ignores partials of the components of $\boldsymbol{a}_t$ 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 (12) 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 [27]. To prevent negative eigenvalues from causing (12) to diverge, SMD uses an extended Gauss-Newton approximation that also admits a fast matrix-vector product. In our case, this is given by

$$\boldsymbol{H}_t \boldsymbol{v}_t \approx \sum_{S_t} \boldsymbol{J}_{\mathcal{F}}^T \boldsymbol{H}_E \boldsymbol{J}_{\mathcal{F}} \boldsymbol{v}_t \quad (13)$$

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 the high-dimensional space of our hand model, the use of constraints is essential. We enforce them by means of a function that after each update (7) maps the parameters back into the feasible region:

$$\boldsymbol{p}_{t+1}^c = \text{constrain}(\boldsymbol{p}_{t+1}) \,. \quad (14)$$

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

$$\boldsymbol{p}_{t+1}^c = \boldsymbol{p}_t^c - \boldsymbol{a}_t \cdot \boldsymbol{g}_t^c \quad \Rightarrow \quad \boldsymbol{g}_t^c = \frac{\boldsymbol{p}_t^c - \boldsymbol{p}_{t+1}^c}{\boldsymbol{a}_t} \,. \quad (15)$$

Using this constrained gradient instead of the ordinary one in Equation (12) enables SMD's step size adaptation machinery to function well in our constrained setting.

4.5 Summary of Gradient Descent Iteration

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

1. calculate the gradient (5),
2. update gradient step sizes (10),
3. update hand model parameters (7),
4. apply hand model constraints (14),
5. calculate constrained gradient (15),
6. calculate the $\boldsymbol{Hv}$ product (13),
7. update SMD's $\boldsymbol{v}$ vector (12).

5 RESULTS

To illustrate the SMD tracker we show results for several 3D hand tracking sequences. The 3D 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. Our experiments have shown that μ is efficient between 0.05 and 0.1, while the parameter λ is typically successful around 1. The following experiments were processed on a Sunfire 1GHz.

5.1 Efficiency of SMD over Classical Methods

First of all, we compare our approach to some major, alternative algorithms (see Subsection 4.1) for multidimensional minimization. The results are presented for a grasping sequence in front of a cluttered background where the hand is rotated and some fingers are hidden due to self-occlusion. The video consisted of 165 frames. The results

of our SMD approach are shown in the top row of Figure 7. Subsequent rows present the results of the Broyden-Fletcher-Goldfarb-Shanno (BFGS) [18], the Gradient Descent (GD) [18], the Powell [18] and the Annealed Particle Filter (APF) [3] algorithms. For BFGS and Powell we used the implementations of the VXL package[3]. 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 optimized 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 can not handle noise well and is mainly designed for unconstrained optimization. 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 like 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 Figure 7 shows, the palm, the thumb and little finger are not always tracked very precisely. Looking at the alternative methods, there is a tradeoff between quality and speed (Table 1).

method	SMD	BFGS	GD	Powell	APF
time [sec/frame]	4.7	8.4	13.5	127.3	131.0

Table 1: Comparison of the computation times.

5.2 Stochastic vs Deterministic

Figure 8 illustrates the effect of deterministically sampled points (bottom row) in comparison to stochastic sampling (top row). This sequence, consisting of 230 frames, shows a situation of occlusion, where parts of the fingers are not observable as they are hidden due to self-occlusion in front of a cluttered background. 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 Figure 9, 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.

[3]http://vxl.sourceforge.net/

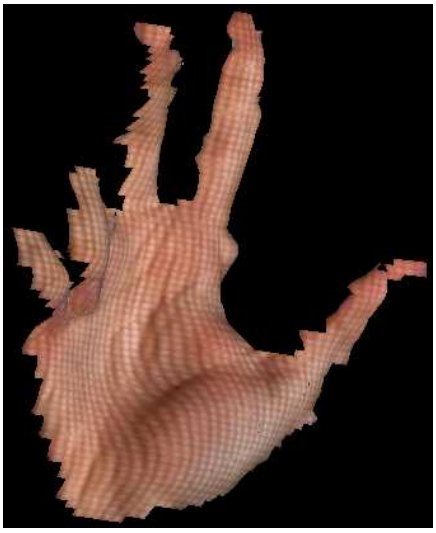


Figure 9: Erroneous depth map from the sequence presented in Figure 8 where the bended fingers are stuck on the palm.

5.3 Animation

Our motion capture is subject to noise due to imperfect depth maps, subsampling and acquisition rate (12.5 frames per second). Therefore, the tracking results need to be cleaned up in order to use them for animation purposes. There is an extensive literature about noise reduction integrated into motion capture system [28, 29, 30]. We chose to use a simple linear adaptive filter which substitutes each data value by a linear combination of its neighboors [28]. With this simple approach, we obtain satisfying results as shown in Figures 10 and 11 which are respectively the virtual replays of the Figures 7 and 8.

6 SUMMARY AND CONCLUSIONS

Model-based tracking of hand articulations is a challenging task due 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 optimization process which is generally a difficult and computationally expensive problem. Our experiments show that the SMD tracker performs robustly and efficiently on rather complex sequences.

We will further investigate the use of multiple hypotheses to improve the handling of occlusions and background clutter. While our approach can overcome local minima, it is not guaranteed that the global minimum will be found. Combining several SMD trackers as smart particles within a particle filter framework is one of the first things on the agenda, in order to achieve such robustness. Using multiple cameras is another interesting extension and could be easily implemented by modifying the error function. Furthermore, adding pseudo joints to the model by using the

method from Mohr and Gleicher [12], would enhance our model, which would also improve the tracking.

Bibliography

[1] Sidenbladh H., Black M.J. and Fleet D.J., 2000, "Stochastic Tracking of 3D Human Figures using 2D Image Motion,". ECCV, pp. 702-718.

[2] Wu Y., Lin J. and Huang T.S., 2001, "Capturing Natural Hand Articulation," ICCV, pp. 426-432.

[3] Deutscher J., Blake A. and Reid I., 2000, "Articulated Body Motion Capture by Annealed Particle Filtering," CVPR, pp. 126-133.

[4] Sminchisescu C. and Triggs B., 2001, "Covariance Scaled Sampling for Monocular 3D Body Tracking," CVPR, pp. 447-454.

[5] Rehg J.M. and Kanade T., 1995, "Model-Based Tracking of Self-Occluding Articulated Objects," ICCV, pp. 612-617.

[6] Isard M. and Blake A., 1998, "Condensation – Conditional Density Propagation for Visual Tracking," IJCV, 29, pp. 5-28.

[7] Schraudolph N.N., 2002, "Fast Curvature Matrix-Vector Products for Second-Order Gradient Descent," Neural Computation, 14, 7, pp. 1723-1738.

[8] Catmull E.E., 1972, "A system for computer generated movies," ACM Annual Conference, pp. 422-431.

[9] Magnenat-Thalmann N., Laperrire R. and Thalman D., 1988 "Jointdependent local deformations for hand animation and object grasping," Graphics Interface, pp. 26-33.

[10] Komatsu K., 1988, "Human Skin Model Capable of Natural Shape Variation," The Visual Computer, 4(3), pp. 265-271.

[11] Lewis J.P., Cordner M. and Fong N., 2000, "Pose space deformations: A unified approach to shape interpolation and skeleton-driven deformation," Annual Conference Series, ACM SIGGRAPH, pp. 165-172.

[12] Mohr A. and Gleicher M., 2003, "Building Efficient, Accurate Character Skins from Examples," Annual Conference Series, ACM SIGGRAPH, pp.562-568.

[13] McDonald J., Toro J., Alkoby K., Berthiaume A., Chomwong P., Christopher J., Davidson M.J., Furst J., Konie B., Lancaster G., Lytinen S., Roychoudhuri L., Sedgwick E., Tomuro N., and Wolfe R., 2000, "An Improved Articulated Model of the Human Hand," 8th Intl. Conf. in Central Europe on Computer Graphics, Visualization and Interactive Digital Media, pp. 306-313.

[14] Lin J., Wu Y. and Huang T.S., 2001, "Modeling Human Hand Constraints," ARL Federated Laboratory 5th Annual Symposium, pp. 105-110.

[15] Buchholz B., Armstrong T.J. and Goldstein S.A., 1992, "Anthropometric Data for Describing the Kinematics of the Human Hand," Ergonomics, 35(2), pp. 261-273.

[16] Koninckx T.P., Griesser A. and Van Gool L., 2003, "Real-Time Range Scanning of Deformable Surfaces by Adaptively Coded Structured Light," 3DIM, pp. 293-300.

[17] Rusinkiewicz S., Hall-Holt O. and Levoy M., 2002 "Real-time 3D model acquisition," Proc. ACM SIGGRAPH, pp. 438-446.

[18] Press W.H., Flannery B.P., Teukolsky S.A. and Vetterling W.T., 1988 "Numerical recipes in C," Cambridge University Press.

[19] Levenberg K., 1944 "A Method for the Solution of Certain Non-Linear Problems in Least Squares," Quarterly Journal of Applied Mathematics, II(2), pp. 164-168.

[20] Marquardt D.W., 1963, "An Algorithm for Least-Squares Estimation of Non-Linear Parameters," Journal of the Society of Industrial and Applied Mathematics, 11(2), pp. 431-441.

[21] Jacobs R.A., 1988, "Increased Rates of Convergence Through Learning Rate Adaptation," Neural Networks, 1, pp. 295-307.

[22] Tollenaere T., 1990, "Super SAB: Fast Adaptive Back Propagation with Good Scaling Properties," Neural Networks, 3, pp. 561-573.

[23] Silva F.M. and Almeida L.B., 1990, "Speeding up backpropagation," Advanced Neural Computers, pp. 151-158.

[24] Almeida L.B., Langlois T., Amaral J.D. and Plakhov A., 1999, "Parameter Adaptation in Stochastic Optimization," On-Line Learning in Neural Networks, Cambridge University Press.

[25] Kivinen J. and Warmuth M.K., 1995, "Additive Versus Exponentiated Gradient Updates For Linear Prediction," Proc. 27th Annual ACM Symposium on Theory of Computing, pp. 209-218.

[26] Sutton R.S., 1992, "Adapting Bias by Gradient Descent: an Incremental Version of Delta-Bar-Delta," Proc. 10th National Conf. Artificial Intelligence, pp. 171-176.

[27] Pearlmutter B.A., 1994, "Fast Exact Multiplication by the Hessian," Neural Computation, 6, 4, pp. 147-160.

[28] Sudarsky S. and House D.H., 2000, "An Integrated Approach towards the Representation, Manipulation and Reuse of Pre-Recorded Motion," Symposium on Computer Animation, pp. 56-61.

[29] Litwinowicz P.C., 1991, "Inkwell: A 2.5D Animation System," ACM SIGGRAPH, pp. 113-122.

[30] Bruderlin A. and Williams L., 1995, "Motion Signal Processing," Computer Graphics, 29, Annual Conference Series, pp. 97-104.

[31] Harmon M.E. and Baird III L.C., "Multi-Player Residual Advantage Learning With General Function Approximation," *Wright Laboratory, WL/AACF*, WL-TR-1065, 1996.

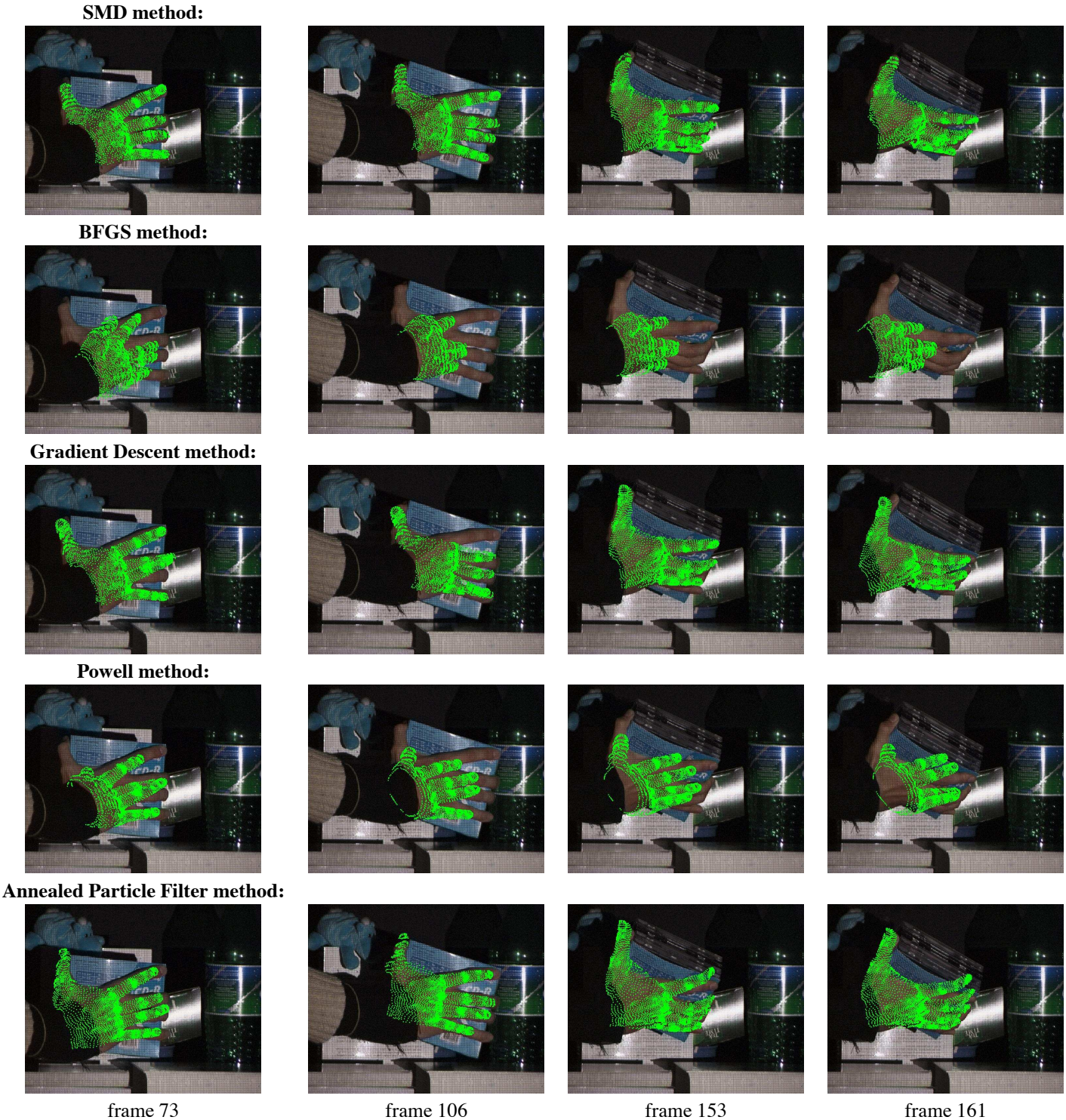


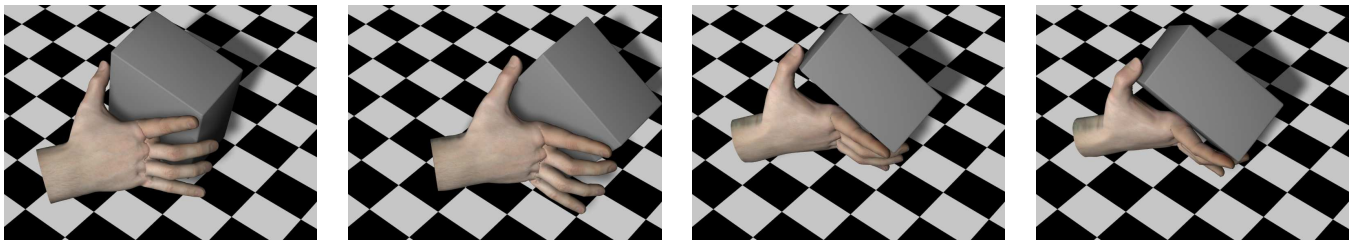
Figure 7: Comparison between the SMD and some standard optimization algorithm. From top to bottom the results of the SMD, BFGS, GD, Powell and APF algorithm are shown. The model is illustated by the visible points on the polygonial surface.

Stochastic Sampling:

Deterministic Sampling:

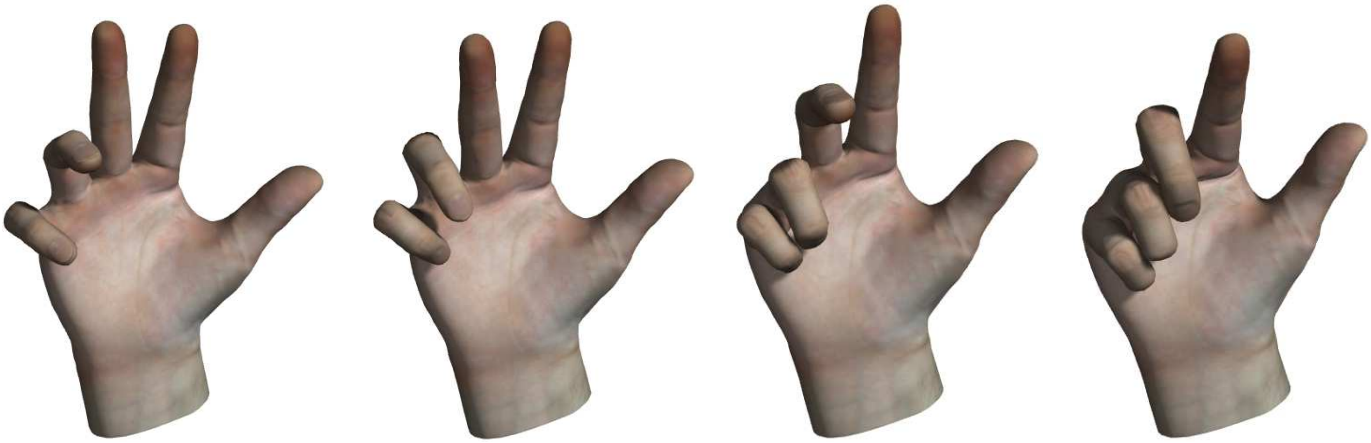
frame 106 frame 136 frame 180 frame 221

Figure 8: Tracking during self-occlusion: stochastic sampling (top row) versus deterministic sampling (bottom row).



frame 73 frame 106 frame 153 frame 161

Figure 10: Virtual replay: reconstructed sequence of the Figure 7, but from a more overhead viewpoint.



frame 106 frame 136 frame 181 frame 221

Figure 11: Virtual replay: reconstructed sequence of the Figure 8.