Available online at www.sciencedirect.com
SCIENCE DIRECT
Image and Vision Computing 25 (2007) 352–364
image AND vision COMPUTING
www.elsevier.com/locate/imavis

Fast stochastic optimization for articulated structure tracking

M. Bray [a,*], E. Koller-Meier [a], N.N. Schraudolph [b], L. Van Gool [a]

[a] Computer Vision Laboratory, Swiss Federal Institute of Technology (ETH), Sternwartstrasse 7, 8092 Zürich, Switzerland
[b] National ICT Australia, Canberra, NSW 2000, Australia

Received 15 October 2004; received in revised form 5 August 2005; accepted 11 October 2005

Abstract

Recently, an optimization approach for fast visual tracking of articulated structures based on stochastic meta-descent (SMD) [7] has been presented. SMD is a gradient descent with local step size adaptation that combines rapid convergence with excellent scalability. Stochastic sampling helps to avoid local minima in the optimization process. We have extended the SMD algorithm with new features for fast and accurate tracking by adapting the different step sizes between as well as within video frames and by introducing a robust cost function, which incorporates both depths and surface orientations. The advantages of the resulting tracker over state-of-the-art methods are supported through 3D hand tracking experiments. A realistic deformable hand model reinforces the accuracy of our tracker.
© 2006 Elsevier B.V. All rights reserved.

Keywords: Stochastic meta-descent; Hand tracking; Deformable hand model

1. Introduction

Tracking of articulated objects is a task as attractive as it is challenging. Attractive because it can be applied to many applications such as motion capture, human–computer interaction (HCI), animation, and medical diagnosis. Indeed, the hand appears to be a natural and efficient vector of communication. The sign language is the most illustrative example. The task is also challenging, especially when the computation time has to be kept low: the more degrees of freedom the object has, the more difficult this task is. Furthermore, in a high dimensional space, many ambiguities may arise.

The introduction of the Condensation algorithm [15] in computer vision proposed wide perspectives concerning the tracking issue as a robust and powerful stochastic sampling approach. However, it turns out that dense sampling becomes infeasible for higher-dimensional state spaces. One has either to lower the dimensionality or to devise schemes that succeed with fewer samples. An action-specific dynamic model allows Sidenbladh et al. [36] to reduce the number of state parameters, though they still need many samples. Wu et al. [40] represent finger joints in a lower-dimensional space by a set of linear manifolds, and then apply a particle filter with importance sampling. Their algorithm performs optimally when the palm is orthogonal to the camera.

Alternatively, one can devise methods that work successfully with a reduced number of samples. Deutscher et al. [12] propose a sparser particle filter based on a simulated annealing algorithm able to track an articulated model in a high dimensional space. Sminchisescu and Triggs [37] combine global sampling with local optimization by gradient descent. However, this approach is rather slow.

The end of the last decade has seen the emergence of data-driven approaches. Their concept is to create a database of different views and configurations of the target with their corresponding features such as silhouette moments [6,33] or feature combinations [2,35,39]. Then, the goal is to explore fast and efficiently this database to find a corresponding match to an input image. This type of method requires a dramatic amount of memory as the dimensionality of the target increases. Furthermore, it is sometimes difficult with just a single image to recover its original configuration when the viewpoint provides poor feature information.

Optimization methods such as gradient descent [5,30], Gauss–Newton method [10], Quasi-Newton methods [24] or trust region method [37] are widely used as well to minimize a cost function while tracking. Recently, we have introduced 'stochastic meta-descent' (SMD) [7] into computer vision for 3D hand tracking and have demonstrated its efficiency for

* Corresponding author. Tel.: +41 1 63 25874; fax: +41 1 63 21199.
E-mail addresses: bray@vision.ee.ethz.ch (M. Bray), ebmeier@vision.ee.ethz.ch (E. Koller-Meier), vangool@vision.ee.ethz.ch (L. Van Gool).

0262-8856/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.imavis.2005.10.009

optimization in high-dimensional spaces. A cost function is minimized by a stochastic gradient descent, with individual step sizes for each dimension that are adapted by a meta-level gradient descent. SMD can naturally incorporate constraints (e.g. anatomical hard constraints) which other optimization techniques find difficult or costly to deal with. Stochasticity in the evaluation of the cost function increases the chance of getting out of local minima. Rehg and Kanade [31] also used gradient descent to minimize the residual between an observed image and overlapping templates describing an articulated object. By comparison, we use 3D rather than 2D video, deal with five rather than two fingers, and employ a more sophisticated gradient descent scheme, a requirement also felt by Rehg and Kanade [31] (p. 616). We extend SMD by adapting the different step sizes between as well as within video frames and by proposing a robust cost function, which includes both depths and surface orientations. We demonstrate it on 3D hand tracking.

The remainder of the paper is organized as follows: Section 2 introduces our 3D hand model and defines the cost function to minimize. Section 3 describes our extended SMD scheme. In Section 4, the incorporation of constraints is discussed and Section 5 explains the inter-frame step size adaptation. Sections 6 and 7 present the results and conclusions, respectively.

2. Model and cost function

2.1. The hand model

The human hand is a complex and highly articulated structure. Many different hand models have been proposed over the last 10 years in order to handle the tracking task.

Three-dimensional models have been widely used with 3D volumetric and kinematic models [11,30,36]. However, in order to support accurate 3D hand tracking (palm skin deformation for instance), the need for more realistic model has increased and many alternatives have emerged. One of the pioneers in realistic human hand modeling was Catmull [9] who proposed a hierarchically structured skin deformation model. 15 years later, Gouret et al. [13] implemented a finite element model to simulate the deformation of objects and human skin while grasping. Kuch and Huang [20] built a 3D model based on cubic B-splines with about 300 control points able to be rendered in real-time. A 3D deformable point distribution model of the hand has been implemented by Heap and Hogg [14] constructed from different human hands. Plaenkers and Fua [27] present an innovative human model where each body part is modeled by soft objects or metaballs, which offer realistic physical deformations.

We use a deformable hand model (see Fig. 1) able to realistically reproduce actual hand shapes. A polygonal skin representation is coupled to an underlying skeleton. This model is able to match most of the hand configurations due to its degrees of freedom (DOFs) and its ability to simulate the skin deformation thanks to the Linear Blend Skinning. We chose this solution because of its fast rendering as the deformations are computed linearly and the simple access to this technology

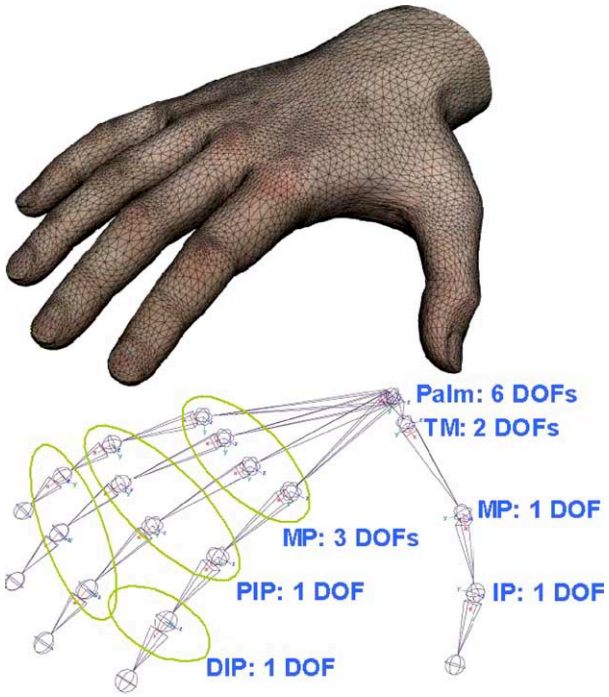

Fig. 1. The hand model as polygonal surface (top). The hand model and its degrees of freedom (bottom), where DIP indicates the 'distal interphalangeal' joints, PIP the 'proximal interphalangeal' joints, MP the 'metacarpophalangeal' joints, IP the 'interphalangeal' and TM the 'trapeziometacarpal' joints.

[18,25]. A more detailed description of the hand model can be found in [7]. Constraints of the human hand reduce the model to 30 degrees of freedom (DOFs) as shown in Fig. 1. The anatomy of the hand imposes limits on the joint angles. We apply such constraints as determined by Lin et al. [23] for our tracking purpose. Additional dependencies between the 'proximal interphalangeal' (PIP) and the 'distal interphalangeal' (DIP) angles (DIP = 2/3 PIP) further reduce our model to 26 DOFs.

2.2. Mapping to camera coordinates

In order to compare a specific hand state to the observed image, the transformation between points on the hand and camera coordinates must be found.

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 encoded as a function $\mathcal{F}$ that, for a given $\boldsymbol{p}$ and intrinsic model parameters $\mathcal{M}$, maps a point $\hat{\boldsymbol{x}}_m$ in 3D hand model coordinates to its (predicted) 3D position $\hat{\boldsymbol{x}}_c$ in camera coordinates:

$$\hat{\boldsymbol{x}}_c = \mathcal{F}(\hat{\boldsymbol{x}}_m, \boldsymbol{p}, \mathcal{M}). \tag{1}$$

For simplicity, a pseudo-orthographic projection is assumed.

2.3. Cost function

Tracking proceeds by matching our 3D hand model against dense 3D measurements extracted at video rate. The input

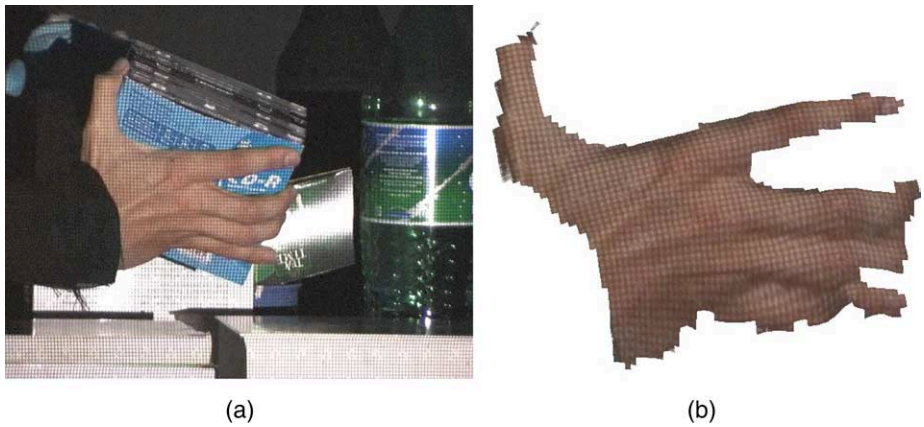

Fig. 2. Input image of the *ShapeSnatcher* (a). Depth map extracted from the scene (a) and masked out by skin color segmentation (b).

speed of our structured light sensor (*ShapeSnatcher* from eyetronics[1]) 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. These said alternative structured light systems exist that does reach the necessary speed [19,34].

ShapeSnatcher is a structured light technology to collect 3D data from a scene. The system extracts a depth map with a single frame while a predefined grid is projected onto an object or a scene. Hence, for a video stream, a full 3D depth map of the scene is acquired for every single frame. The camera shots the image from a slightly different point of view than the projector (a few degrees). The depth map is defined according to the deformation of one or more grid projections [29]. Fig. 2 shows the data provided by *ShapeSnatcher*. The left picture (a) is a sample from a tracking video and by looking carefully; one can observe the grid of the structured light system projected on the scene. The right picture (b) presents the resulting depth map of (a), masked out by skin color segmentation: the hand's edges are noisy due to the fact that the projected grid is heavily deformed on all edges. Furthermore, one can notice that the last phalanx of the thumb is not extracted and the little finger presents wrong 3D data as it appears 'pushed' into the scene.

Three-dimensional video has been as well exploited to perform articulated structure tracking so as to overcome the ambiguities due to 2D information. Delamarre and Faugeras [11] estimate the motion of articulated objects like human bodies and hands against cluttered background. With the use of three calibrated cameras, they can extract range maps from the scene. They aim at minimizing the distance between the surface of the 3D reconstruction and the surface of their 3D model. This distance is evaluated by physical forces (normal and spring-like forces). The minimization of the cost function is then performed by solving dynamical equations of motion. Plaenkers and Fua [27] introduce a cost function, relying on depth information provided by three cameras as well, which is minimized by the Levenberg–Marquardt algorithm. Lin [24] proposes a real-time implementation for tracking an arm in dense range maps. The articulated model is represented by a set of planar patches bounded by the convex hull of two circles. The cost function describes the distance between the pixel depth inside each limb model and the corresponding predicted depth. The optimization is performed by the Broyden–Fletcher–Golfarb–Shanno minimization. However, the author reports the inaccuracy of such a model.

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 iterative closest point (ICP) [4] 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. $\hat{\boldsymbol{x}}'_c = (\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}, \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}))$. This is illustrated in Fig. 3. Interpolation fills in small holes. Moreover, the difference between the observed normal $\boldsymbol{n}$ and the predicted normal $\hat{\boldsymbol{n}}$ at these points is also considered.

If the closest point on this plane is $\boldsymbol{x}_c$, the tracker minimizes the following sum-squared cost function over a sample set S_i

$$E = \sum_{S_i} \frac{1}{2} \left(\|\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c\|^2 + k\|\hat{\boldsymbol{n}} - \boldsymbol{n}\|^2 \right), \tag{2}$$

where $\|\cdot\|$ denotes the L_2-norm. A value of $k=3$ was found to provide a robust cost function. Results in Section 6 show that

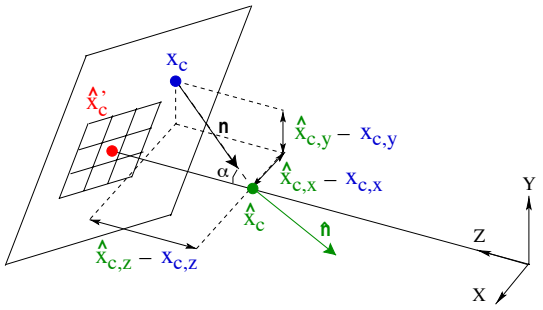

Fig. 3. The calculation of the error function. An approximation to the closest point on the surface is calculated by projecting the model point $\hat{\boldsymbol{x}}_c$ onto the tangential plane at $\hat{\boldsymbol{x}}'_c = (\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}, \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x}, \hat{\boldsymbol{x}}_{c,y}))$. This yields the approximation $\boldsymbol{x}_c$. $\mathcal{Z}$ is the depth map provided by the 3D sensor in camera coordinates. Furthermore, the difference between the observed normal $\boldsymbol{n}$ and the predicted normal $\hat{\boldsymbol{n}}$ is taken into account of the cost function.

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

incorporating the normal increases the robustness of the SMD tracker. The cost function E is evaluated by considering only a rather limited set of points, 45 in our current implementation, to speed up the tracking (see Section 2.4).

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

$$\boldsymbol{n}* = \left[\left.\frac{\partial \mathcal{Z}}{\partial x}\right|_{\hat{\boldsymbol{x}}_c}, \left.\frac{\partial \mathcal{Z}}{\partial y}\right|_{\hat{\boldsymbol{x}}_c}, -1\right]^{\mathrm{T}} \Rightarrow \boldsymbol{n} = \frac{\boldsymbol{n}^*}{||\boldsymbol{n}^*||}, \tag{3}$$

where T denotes the matrix transpose.

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

$$\begin{aligned}
\boldsymbol{x}_c &= \hat{\boldsymbol{x}}_c + [(\hat{\boldsymbol{x}}_c - \hat{\boldsymbol{x}}_c)\cdot\boldsymbol{n}]\boldsymbol{n} \\
\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c &= \left[\left(\begin{pmatrix}\hat{x}_{c,x}\\ \hat{x}_{c,y}\\ \hat{x}_{c,z}\end{pmatrix} - \begin{pmatrix}\hat{x}_{c,x}\\ \hat{x}_{c,y}\\ \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y})\end{pmatrix}\right)\cdot\boldsymbol{n}\right]\boldsymbol{n} \\
&= \left[\begin{pmatrix}0\\ 0\\ \hat{\boldsymbol{x}}_{c,z} - \mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y})\end{pmatrix}\cdot\boldsymbol{n}\right]\boldsymbol{n} \\
&= \frac{[\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}]\boldsymbol{n}*}{||\boldsymbol{n}*||^2}.
\end{aligned} \tag{4}$$

Then, we derive the components of (2) from (4):

$$\begin{aligned}
\hat{\boldsymbol{x}}_{c,x} - \boldsymbol{x}_{c,x} &= \frac{[\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}]\boldsymbol{n}_x^*}{||\boldsymbol{n}*||^2} \\
&= \frac{\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y}) - \hat{\boldsymbol{x}}_{c,z}}{||\boldsymbol{n}*||^2}\frac{\partial \mathcal{Z}}{\partial x},
\end{aligned} \tag{5}$$

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

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

The gradients of the depth map $\mathcal{Z}$ are computed numerically, using 3×3 Sobel masks.

Our scheme requires the Jacobian of E, $\boldsymbol{J}_E$, and its Hessian $\boldsymbol{H}_E$. In order to compute $\boldsymbol{J}_E$ and $\boldsymbol{H}_E$, it is important to note that:

- $\partial(\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c)/\partial\hat{\boldsymbol{n}} = \partial(\hat{\boldsymbol{n}} - \boldsymbol{n})/\partial\hat{\boldsymbol{x}}_c = 0$ as we do not differentiate through F but only through E. The inter-dependence between $(\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c)$ and $(\hat{\boldsymbol{n}} - \boldsymbol{n})$ is taken into account while calculating the gradient of E. Indeed, the gradient of E is calculated by backward chaining of derivatives (cf. Eq. (16)).
- $\partial(\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y}) - \boldsymbol{x}_{c,z})/\partial\hat{\boldsymbol{x}}_c = \boldsymbol{n}*$
- As $\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y})$ is quite noisy, we assume that the depth map has no second derivative, i.e. $\partial^2\mathcal{Z}(\hat{\boldsymbol{x}}_{c,x},\hat{\boldsymbol{x}}_{c,y})/\partial\hat{\boldsymbol{x}}^2 = 0$.

In order to evaluate the derivatives in $\boldsymbol{J}_E$ and $\boldsymbol{H}_E$, we need to know how a change in $\hat{x}_c$ influences $\boldsymbol{x}_c$. In order to simplify notations, vector coordinates will be written as $\boldsymbol{x}_c = [x_c, y_c, z_c]^{\mathbf{T}}$ and $\hat{\boldsymbol{x}}_c = [\hat{x}_c, \hat{y}_c, \hat{z}_c]^{\mathrm{T}}$. The components of the normal $\boldsymbol{n}$ will be denoted as $\boldsymbol{n} = [n_x, n_y, n_z]^{\mathrm{T}}$. Fig. 4 shows a translation from $\hat{\boldsymbol{x}}_c^a$ to $\hat{\boldsymbol{x}}_c^b$ projected onto the plane Π as $\boldsymbol{x}_c^a$ to $\boldsymbol{x}_c^b$. Note that the tangent plane Π is assumed to have remained fixed (thereby discarding surface curvatures as stated before). One can therefore deduce:

$$\boldsymbol{x}_c^b - \boldsymbol{x}_c^a = \hat{\boldsymbol{x}}_c^b - \hat{\boldsymbol{x}}_c^a - ((\hat{\boldsymbol{x}}_c^b - \hat{\boldsymbol{x}}_c^a)\cdot\boldsymbol{n})\boldsymbol{n}, \quad = \Delta\hat{\boldsymbol{x}}_c - (\Delta\hat{\boldsymbol{x}}_c\cdot\boldsymbol{n})\boldsymbol{n}. \tag{8}$$

Given the assumption $\partial(\hat{\boldsymbol{n}} - \boldsymbol{n})/\partial\hat{\boldsymbol{x}}_c = 0$, the first derivative is calculated as

$$\frac{\partial E}{\partial \hat{\boldsymbol{x}}_c} = \frac{\partial\left[\frac{1}{2}((\hat{x}_c - x_c)^2 + (\hat{y}_c - y_c)^2 + (\hat{z}_c - z_c)^2)\right]}{\partial\hat{\boldsymbol{x}}_c}. \tag{9}$$

To obtain the first x component of Eq. (9), we apply the chain rule to get

$$\frac{\partial\left[\frac{1}{2}(\hat{x}_c - x_c)^2\right]}{\partial\hat{x}_c} = (\hat{x}_c - x_c)\left(1 - \frac{\partial x_c}{\partial\hat{x}_c}\right). \tag{10}$$

Using Eq. (8) we get

$$\begin{aligned}
\frac{\partial x_c}{\partial\hat{x}_c} &= \lim_{\Delta\hat{x}_c\to 0}\frac{(\Delta\hat{x}_c - (\Delta\hat{\boldsymbol{x}}_c\cdot\boldsymbol{n})n_x)}{\Delta\hat{x}_c} \\
&= \lim_{\Delta\hat{x}_c\to 0}\left(1 - \frac{(\Delta\hat{x}_c n_x + \Delta\hat{y}_c n_y + \Delta\hat{z}_c n_z)n_x}{\Delta\hat{x}_c}\right) \\
&= 1 - n_x^2.
\end{aligned} \tag{11}$$

By calculating the other components similarly, we obtain

$$\begin{aligned}
\frac{\partial E}{\partial\hat{x}_c} &= (\hat{x}_c - x_c)n_x^2 + (\hat{y}_c - y_c)n_x n_y + (\hat{z}_c - z_c)n_x n_z \\
\frac{\partial E}{\partial\hat{y}_c} &= (\hat{x}_c - x_c)n_x n_y + (\hat{y}_c - y_c)n_y^2 + (\hat{z}_c - z_c)n_y n_z \\
\frac{\partial E}{\partial\hat{z}_c} &= (\hat{x}_c - x_c)n_x n_z + (\hat{y}_c - y_c)n_y n_z + (\hat{z}_c - z_c)n_z^2,
\end{aligned} \tag{12}$$

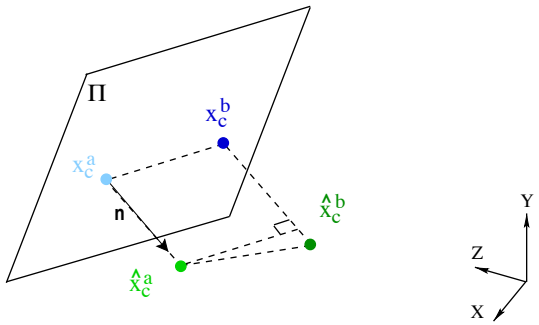

Fig. 4. Sketch of the assumed displacement of $\boldsymbol{x}_c$ given a displacement of $\hat{\boldsymbol{x}}_c$.

and for the second derivatives:

$$\begin{aligned}\frac{\partial^2 E}{\partial^2 \hat{x}_c} &= n_x^2(1-(1-n_x^2)) + n_x n_y n_x n_y + n_x n_z n_x n_z = n_x^2 \\ \frac{\partial^2 E}{\partial \hat{x}_c \partial \hat{y}_c} &= n_x^2 n_x n_y + n_x n_y n_y^2 + n_x n_z n_z n_y = n_x n_y \\ \frac{\partial^2 E}{\partial \hat{x}_c \partial \hat{z}_c} &= n_x^2 n_x n_z + n_x n_y n_y n_z + n_x n_z n_z n_z = n_x n_z.\end{aligned} \tag{13}$$

By symmetry, we have $\partial^2 E/\partial \hat{\boldsymbol{x}}_c^2 = \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}$. In conclusion, we have the following Jacobian and Hessian

$$\begin{aligned}\boldsymbol{J}_E &= \frac{\partial E}{\partial(\hat{\boldsymbol{x}}_c, \hat{\boldsymbol{n}})} = \begin{bmatrix} \dfrac{\partial}{\partial \hat{\boldsymbol{x}}}\left(\dfrac{1}{2}(\hat{\boldsymbol{x}}-\boldsymbol{x})^2\right) \\ \dfrac{\partial}{\partial \hat{\boldsymbol{n}}}\left(\dfrac{1}{2}k(\hat{\boldsymbol{n}}-\boldsymbol{n})^2\right) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}}(\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c) \\ k(\hat{\boldsymbol{n}}-\boldsymbol{n}) \end{bmatrix}\end{aligned} \tag{14}$$

$$\begin{aligned}\boldsymbol{H}_E &= \frac{\partial^2 E}{\partial(\hat{\boldsymbol{x}}_c, \hat{\boldsymbol{n}})^2} = \begin{bmatrix} \dfrac{\partial^2 E}{\partial \hat{\boldsymbol{x}}_c^2} & \dfrac{\partial}{\partial \hat{\boldsymbol{n}}_c}(\hat{\boldsymbol{x}}_c - \boldsymbol{x}_c) \\ \dfrac{\partial}{\partial \hat{\boldsymbol{x}}_c} k(\hat{\boldsymbol{n}}_c - \boldsymbol{n}_c) & \dfrac{\partial}{\partial \hat{\boldsymbol{n}}_c} k(\hat{\boldsymbol{n}}_c - \boldsymbol{n}_c) \end{bmatrix} \\ &= \begin{bmatrix} \boldsymbol{n}\boldsymbol{n}^{\mathrm{T}} & 0 \\ 0 & k\mathrm{Id} \end{bmatrix}\end{aligned} \tag{15}$$

where **Id** represents the identity matrix.

2.4. Stochastic sampling

In order to preserve a competitive time processing, the tracker cannot afford to sample the complete depth map, which represents between 5000 and 7000 pixels according to the hand state. Therefore, E is evaluated by considering only a rather limited set of points, which is changed in each iteration during the optimization step to offer a better robustness. Indeed, the stochasticity allows a faster averaging of the probability density function than with a deterministic sampling.

The question is now to define the minimum number of sample points, which will offer the best trade-off between speed and accuracy. Hence, we run our tracker for the same sequence with different number of sample points in two modes: the first one is to sample randomly a fixed number of points on the visible part of the hand. The second is too select randomly a fixed number (depending of their visibility) of sample points on each phalanx and the palm to guarantee that each dimension of the search space will be represented. We will name this mode semi-random sampling. Every five iterations, a visibility algorithm is performed (depth-buffer algorithm) and if some phalanges are occluded for instance, the sampled points dedicated to them are randomly chosen on the visible part of the hand.

Fig. 5 presents on the left side (a) the quantitative result of this experiment and the right side (b) shows an example of semi-random sampling with 2 points per phalanx and 15 points on the palm (or the back of the palm according to the visibility).

One can observe on Fig. 5(a) that a semi-random sampling converges faster than a random sampling. It can easily be explained by the fact that to perform robustly, one needs to guarantee that at each iteration, every dimension of the state space is represented by the samples. Then, we can notice that the relationship between the number of sample points and the

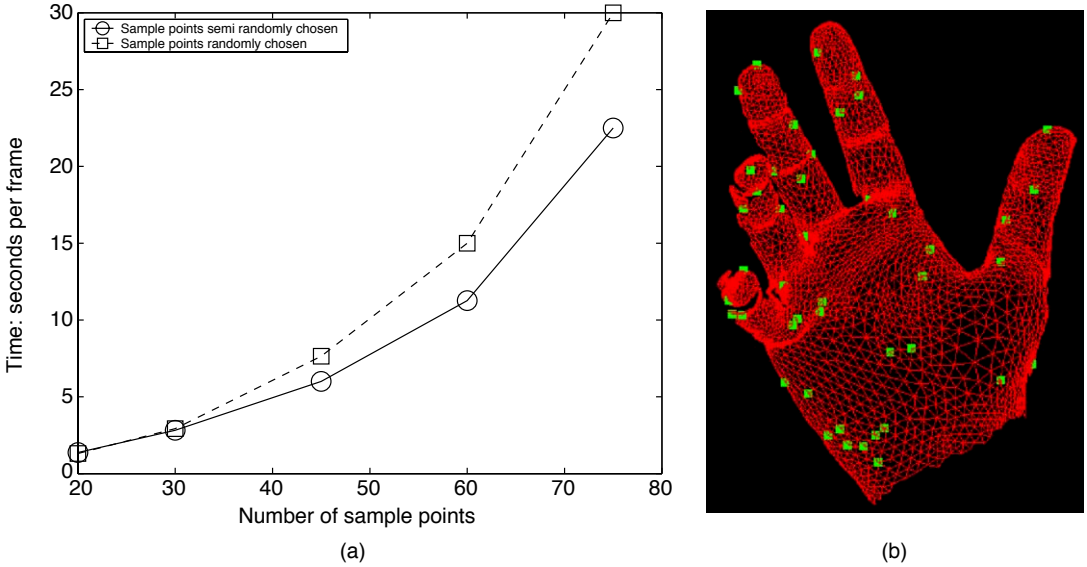

Fig. 5. On the left (a), evolution of the tracker's speed according to the number of sample points and the sampling mode. On the right (b), illustration of a random subsampling performed on the hand model with 2 points per phalanx and 15 points for the visible part of the palm. When a hand part is not visible, the points expected on it are randomly chosen on any other visible part.

speed of the tracker is closer to an exponential model than a linear model.

Fig. 6 presents the qualitative results of Fig. 5(a): they illustrate the influence of the sampling on the accuracy and robustness of the tracker. By looking at the second and fourth rows of Fig. 6, which shows the effect of pure random sampling, one can notice that while bending the fingers, the model either loses the target (20 points), or has a delay in comparison with the observations. In this sampling mode, it turns out that less than 30 points are not sufficient to correctly explore a 26 dimensional-space. From 45 points on, the tracker seems to stabilize and recovers well the observations. However, as seen in Fig. 5(a), with the number of sample points increasing, the tracker's speed decreases dramatically. The third and fifth rows of Fig. 6 demonstrate the quality of tracking by using a semi-random sampling where points are randomly spread on every single visible phalanx and the visible part of the palm. For the same number of points the semi-random sampling appears to be more robust than a random sampling and converges faster as shown in Fig. 5(a). As for the random sampling, the tracker becomes fully effective from 45 points on. Therefore, we choose to use the semi-random sampling approach with 2 points per phalanx and 15 points on the palm.

3. The SMD algorithm

Nonlinear optimization problems are often solved by second-order gradient techniques such as the Levenberg–Marquardt algorithm [21,26] or truncated Quasi–Newton methods like the

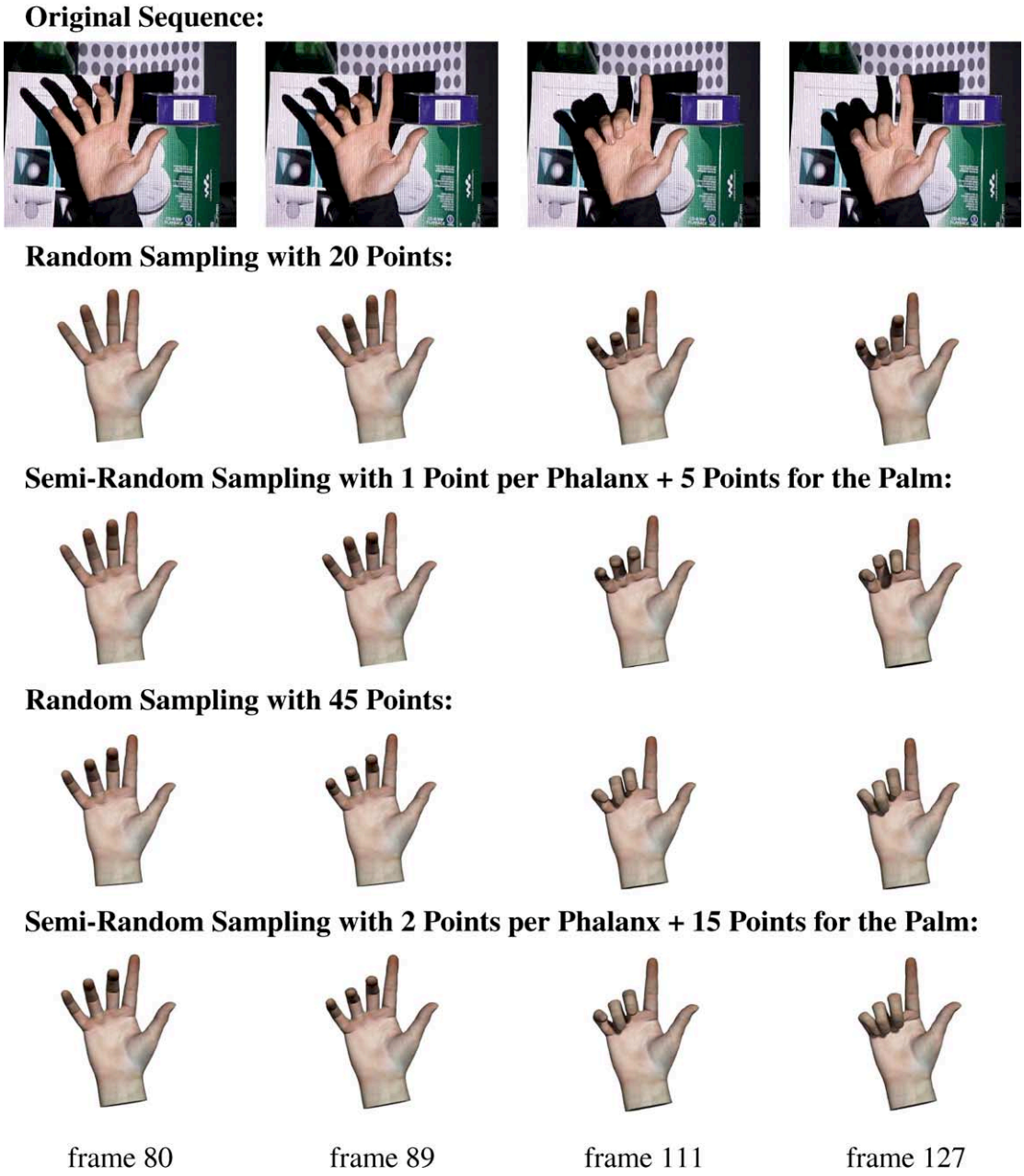

Fig. 6. Original sequence and 3D reconstructions of the experience presented in Fig. 5(a).

Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [28]. These techniques update the model parameters in large steps, each of which is relatively expensive to compute. This makes it hard to enforce constraints on the parameters: the farther a single step takes 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, which are computationally too expensive for our purposes. Conjugate gradient (CG) techniques [28] are computationally cheaper—linear cost per iteration—but 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. CG also does not tolerate the noise inherent in our approximation of the gradient by sampling, and converges poorly on highly nonlinear problems.

SMD's strength lies in three aspects [7]. Firstly, the use of stochasticity lets the method escape from local minima more easily. Secondly, the use of separate, adaptive step sizes per dimension makes it more efficient in the case of ill-conditioned systems whose energy landscapes show long, narrow valleys. Thirdly, it updates parameters while taking account of the past history of step sizes, and thus is able to capture long-range effects missed by other algorithms. This dampens erratic variations and increases the method's efficiency.

Here, we give a concise overview of SMD. If $\boldsymbol{g}_i$ is the gradient (of $E \circ \mathcal{F}$) at iteration step i, i.e.

$$\boldsymbol{g}_i \equiv \frac{\partial E}{\partial \boldsymbol{p}_i} = \sum_{S_i} \boldsymbol{J}_{\mathcal{F}}^{\mathrm{T}} \boldsymbol{J}_E^{\mathrm{T}}, \tag{16}$$

the parameter vector $\boldsymbol{p}_i$ is updated via

$$\boldsymbol{p}_{i+1} = \boldsymbol{p}_i - \boldsymbol{a}_i \otimes \boldsymbol{g}_i, \tag{17}$$

where $\otimes$ denotes the Hadamard (i.e. component-wise) product, $\boldsymbol{J}_{\mathcal{F}}$ the Jacobian of the function $\mathcal{F}$, $\boldsymbol{J}_E$ the Jacobian of the function E and T the matrix transpose. The vector $\boldsymbol{a}$ of local step sizes is in effect a diagonal conditioner for the gradient system. S_i represents the sample set, which is stochastically changed for each iteration step i.

A classical method in order to adapt $\boldsymbol{a}$ during the optimization is to modify the step sizes according to the sign of the autocorrelation of the gradient ($\boldsymbol{g}_i \otimes \boldsymbol{g}_{i-1}$) [16]: in the case where the gradient oscillates (i.e. $\boldsymbol{g}_i \cdot \boldsymbol{g}_{i-1} < 0$), the step sizes are diminished and vice versa. This approach should not be used in combination with stochastic gradient estimates however, since the sign-based stochastic adaptation method tends to fail frequently [1]. 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 [16]. To allow $\boldsymbol{a}$ to remain strictly positive, the meta-level descent is best performed on $\ln \boldsymbol{a}$ [17]. Assuming that each element of $\boldsymbol{a}$ affects the cost function E only through the corresponding element of $\boldsymbol{p}$, such that the chain rule can be used, we obtain (still using the Hadamard product)

$$\ln \boldsymbol{a}_i = \ln \boldsymbol{a}_{i-1} - \mu \frac{\partial E}{\partial \boldsymbol{p}_i} \otimes \frac{\partial \boldsymbol{p}_i}{\partial \ln \boldsymbol{a}_{i-1}} = \ln \boldsymbol{a}_{i-1} + \mu \boldsymbol{g}_i \otimes \boldsymbol{v}_i, \tag{18}$$

where μ is a scalar meta-step size, and $\boldsymbol{v}_i$ characterizes the dependence of parameters on their step sizes. From (17), we find that

$$\boldsymbol{v}_{i+1} \equiv -\frac{\partial \boldsymbol{p}_{i+1}}{\partial \ln \boldsymbol{a}_i} = \boldsymbol{a}_i \otimes \boldsymbol{g}_i \tag{19}$$

which inserted into (18), gives us the familiar autocorrelation $\boldsymbol{g}_i \otimes \boldsymbol{g}_{i-1}$ of the gradient, now without the problematic sign function [1]. Finally, exponentiating (18) gives

$$\begin{aligned} \boldsymbol{a}_i &= \boldsymbol{a}_{i-1} \otimes \exp(\mu \boldsymbol{g}_i \otimes \boldsymbol{v}_i) \\ &\approx \boldsymbol{a}_{i-1} \otimes \max\left(\frac{\mathbf{1}}{\mathbf{2}}, \mathbf{1} + \mu \boldsymbol{v}_i \otimes \boldsymbol{g}_i\right), \end{aligned} \tag{20}$$

where $\mathbf{1/2}$ and $\mathbf{1}$ are vectors where all the elements are, respectively, equal to 1/2 and 1. 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.

The simple definition (19) for $\boldsymbol{v}$ has the severe disadvantage that it fails to take into account long-term dependencies of parameter values $\boldsymbol{p}$ on step sizes $\boldsymbol{a}$. Ideally, one would like to model the effect of the current step size on future weights. In contrast, [38] 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 *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:

$$\boldsymbol{v}_{i+1} \equiv -\sum_{k=0}^{\infty} \lambda^k \frac{\partial \boldsymbol{p}_{i+1}}{\partial \ln \boldsymbol{a}_{i-k}}. \tag{21}$$

The factor $0 \leq \lambda \leq 1$ governs the time scale over which long-term dependencies are taken into account. Inserting (17) into (21) gives

$$\begin{aligned} \boldsymbol{v}_{i+1} &= -\sum_{k=0}^{\infty} \lambda^k \frac{\partial \boldsymbol{p}_i}{\partial \ln \boldsymbol{a}_{i-k}} + \sum_{k=0}^{\infty} \lambda^k \frac{\partial (\boldsymbol{a}_i \otimes \boldsymbol{g}_i)}{\partial \ln \boldsymbol{a}_{i-k}} \\ &\approx \lambda \boldsymbol{v}_i + \boldsymbol{a}_i \otimes \boldsymbol{g}_i + \boldsymbol{a}_i \otimes \left[\frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{p}_i^{\mathrm{T}}} \sum_{k=0}^{\infty} \lambda^k \frac{\partial \boldsymbol{p}_i}{\partial \ln \boldsymbol{a}_{i-k}} \right] \\ &= \lambda \boldsymbol{v}_i + \boldsymbol{a}_i \otimes (\boldsymbol{g}_i - \lambda \boldsymbol{H}_i \boldsymbol{v}_i) \end{aligned} \tag{22}$$

where $\boldsymbol{H}_i$ denotes the instantaneous Hessian at iteration i. For the sake of simplicity, (22) 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 (22) 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 [32]. To prevent negative eigenvalues from causing (22) to diverge, SMD uses an extended Gauss–Newton approximation that also admits a fast matrix-vector product. In our case, this is given by

$$\boldsymbol{H}_i\boldsymbol{v}_i \approx \sum_{S_i} \boldsymbol{J}_{\mathcal{F}}^{\mathrm{T}}\boldsymbol{H}_E\boldsymbol{J}_{\mathcal{F}}\boldsymbol{v}_i \tag{23}$$

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

4. Incorporation of constraints

The classical way to perform a constrained optimization is by the use of the Lagrange multipliers [3] although computationally expensive. An alternative proposed by Wu et al. [40] has been to learn the anatomical constraints of a hand, based on natural motion via a cyberglove. The joint space of the hand was consequently reduced to a seven-dimensional subspace approximated by the union of basis configurations. Then, using importance sampling, the learned space was sampled to evaluate the hand state.

The introduction of the hand model constraints into the optimization is most beneficial in the high-dimensional space that we have to explore. We enforce them by means of a function that after each update (17) maps the parameters back into the feasible region:

$$\boldsymbol{p}_{i+1}^{c} = \mathrm{constrain}(\boldsymbol{p}_{i+1}). \tag{24}$$

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

$$\boldsymbol{p}_{i+1}^{c} = \boldsymbol{p}_i^{c} - \boldsymbol{a}_i \otimes \boldsymbol{g}_i^{c} \Rightarrow \boldsymbol{g}_i^{c} = \frac{\boldsymbol{p}_i^{c} - \boldsymbol{p}_{i+1}^{c}}{\boldsymbol{a}_i}. \tag{25}$$

By using this constrained gradient instead of the usual one in Eq. (22), we can get SMD's step size adaptation machinery to work well for constrained optimization.

5. Inter frame step size adaptation

In a tracking context, SMD has to minimize the cost function several times for subsequent frames. Resetting all the parameters to their initial values would be undesirable. A new extension to SMD is to let the system benefit from experience gathered during the previous frame. As the first step of intra-frame SMD gives information about whether the different step sizes in $\boldsymbol{a}_0$ are too small or too large and suggests improved step sizes $\boldsymbol{a}_1$ to start intra-frame optimization within that frame, we take these improved values as appropriate initial step sizes $\boldsymbol{a}_0$ for the intra-frame optimization in the next frame. With a good approximation of the initial step sizes, convergence is reached with less iteration. Fig. 7 shows the advantageous effect of using this information. It compares the accumulated number of iterations with (solid line) and without (dashed line) this initialization based on the previous frame. Without, the step sizes are reset at each frame to the same values. The inter-frame adaptation accelerates the tracking by 19%.

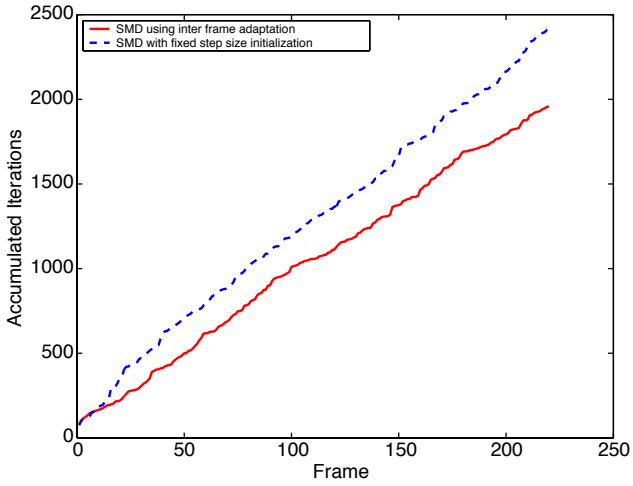

Fig. 7. Comparison of the accumulated number of iterations when tracking with or without inter-frame step size adaptation. The solid line is with inter-frame initialization, the dashed line when step sizes are simply reset at the beginning of each frame.

This extended version of SMD can be summarized as follows:

- LOOP $t{:=}1$ TO last frame
  - $\boldsymbol{v}_0{:=}\boldsymbol{0}$; $\boldsymbol{a}_0{:=}\boldsymbol{a}^*$; $\boldsymbol{p}_0{:=}\mathit{InitialState}$
  - LOOP $i{:=}1$ TO convergence
    (1) pick sample points S_i;
    (2) calculate $\boldsymbol{g}_i$ (16) and $\boldsymbol{a}_i$ (20);
    – IF $i{=}1$ THEN $\boldsymbol{a}^*{:=}\boldsymbol{a}_1$;
    (3) calculate $\boldsymbol{p}_{i+1}$ (17),
    (4) calculate $\boldsymbol{p}_{i+1}^{c}$ (24),
    (5) calculate $\boldsymbol{g}_i^{c}$ (25), $\boldsymbol{H}_i\boldsymbol{v}_i$ (23), $\boldsymbol{v}_{i+1}$ (22).

The initial pose $\boldsymbol{p}_0$ of the hand is determined by manual alignment in the first frame.

6. Results

The SMD tracker's performance is illustrated on 3D hand data, obtained with a structured light sensor yielding depth maps of 720×576 pixels at 12.5 frames per second. The processing was carried out on a Sunfire 1 GHz PC. The first experiment presented in Fig. 8 highlights the importance of stochasticity. Fig. 8 shows the evolution of the SMD tracker, while initialized fairly far from its target, projected on the translation in X and rotation in Z of the palm. As shown the tracker recovers the target in seven iterations. The landscape of the displayed function is changing over time due to the stochastic sampling. The global extremum remains naturally

nearly the same, however, the local minima are changing constantly. The SMD tracker can therefore avoid spurious minima, increasing the probability to match correctly the target. In comparison, Fig. 9 presents for the same sequence, the results obtained by gradient descent and a deterministic sampling approach. The gradient descent method needs 16

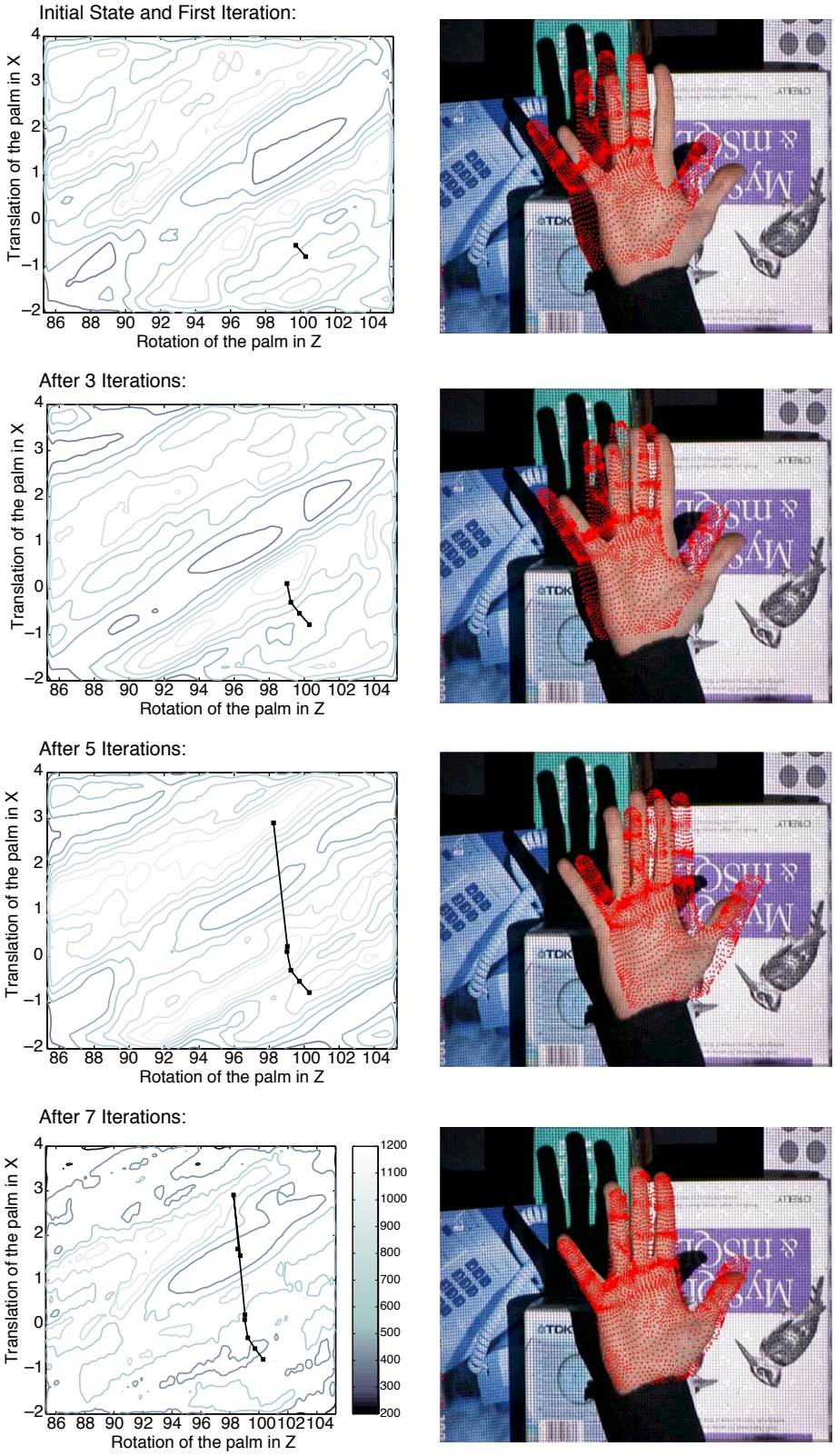

Fig. 8. Evolution of the SMD algorithm while tracking. The top row shows this evolution projected on the translation in X and rotation in Z of the palm and the bottom rows present the resulting sequence.

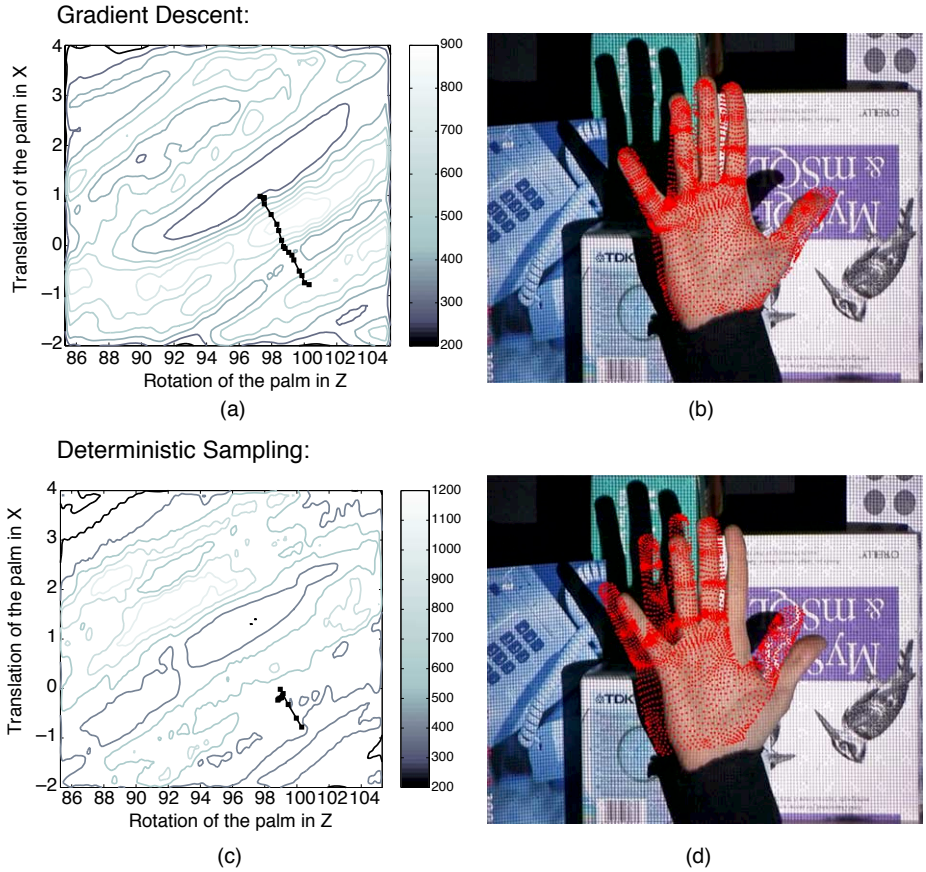

Fig. 9. Evolution of the gradient descent algorithm ((a) and (b)) and a deterministic sampling ((c) and (d)) while tracking the same sequence as in Fig. 8. (b) Is the final result of the gradient descent after 16 iterations: the hand model is slightly shifted from the target. (d) Is the final result of the deterministic sampling after nine iterations: this approach clearly diverges from the correct solution.

iterations to reach an acceptable match but as one can observe, the hand model is shifted from the target providing a worse match than SMD. The deterministic sampling approach, as expected, fails after a few iteration. If more points were used to sample the hand model, the deterministic sampling would be less sensitive to local minima but this would increase the processing time.

Fig. 10 illustrates the good performance of the SMD tracker when compared to conventional optimization schemes such as gradient descent (GD) [28] or Powell [28], as well as to the state-of-the-art annealed particle filter (APF) [12]. Furthermore, the second row of Fig. 10 shows the SMD tracker using a cost function E defined by the sample positions only, i.e. without surface orientation. All methods were running on exactly the same 3D input data and were optimizing the same cost function. The parameters in all methods were chosen carefully in order to optimize their results. For Powell we used the implementations of the VXL package[2]. GD was applied by using SMD and setting μ and λ to zero. For APF, we had to use our own implementation. This may not have been maximally optimized for speed, but neither is our current implementation of SMD. The first two rows in Fig. 10 show that the incorporation of surface orientations in E increases the robustness of the SMD tracker. With only the position used in E, the tracker gets trapped in a local minimum more easily (in this example, the thumb gets stuck to the palm). In this experiment—and several additional, similar experiments—SMD performed best, not only in terms of accuracy but also in terms of computation time. The speeds given in seconds/frame for the experiment in Fig. 10 are presented in Table 1.

The GD approach is as fast as SMD, but falls in a local minimum and is unable to recover. Powell's method is rather slow as it does not use gradient information and it loses the correct solution early on. Besides the SMD algorithm, APF provides the most convincing results, because it is more apt to find the global minimum. But, as shown in Fig. 10, in the last frame, the index finger is unable to bend correctly and loses the target. SMD with a cost function that uses position and orientation differences seems to offer the best compromise between speed and accuracy.

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

Fig. 10. Comparison between SMD and alternative optimization algorithms. From top to bottom: the results of the SMD using a cost function E evaluating the distances and the differences in surface orientations, SMD using a cost function E only evaluating the distances, gradient descent, Powell and annealed particle filter (APF). The model is visualized as the vertices of the skin polygonal representation. The pictures have been cropped for better visibility, but see Fig. 11 for examples of similar, complete frames.

Fig. 11 gives an example with serious self-occlusion, where the SMD tracker does a good job in figuring out the overall rotation that the hand makes. Fig. 12 shows examples from an experiment where the word 'FLY' was formed in American sign language (ASL). As a matter of fact, the hand pattern formed in Fig. 10 corresponds to the letter 'A'. Although SMD shows a rather good performance, we have observed some recurring problems in our overall set of experiments. The most prominent one is that sometimes the thumb tip was still not tracked well due to the paucity of the depth map in that area. The thumb is more prone to errors in this regard, because of its wider range of motions and its outspoken independence of the other fingers.

Some more results can be found in [7].

Table 1
Comparison of the computation time of the Fig. 10

Method	SMD	GD	Powell	APF
Time (s/frame)	3	3	232	114

Fig. 11. Hand tracking with self-occlusions. Far right: 3D model for frame 103 (top view).

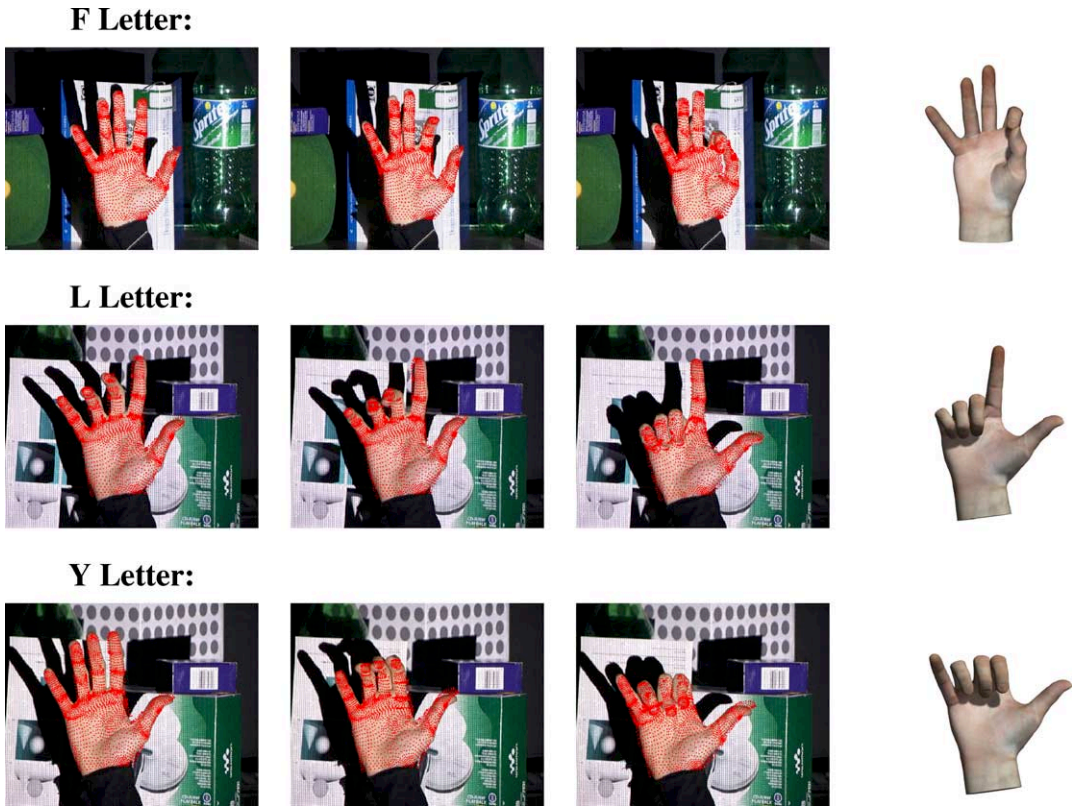

Fig. 12. Spelling the word 'FLY' in American sign language with SMD: top row represents the 'F' letter, the second row the 'L' letter and the last row the 'Y' letter. The last column is the 3D reconstruction from the third one.

7. Conclusions and future work

We have presented a novel, extended SMD tracker, which is based on a rapid stochastic gradient descent approach with adaptive step sizes. The inclusion of constraints that can be imposed on the many DOFs of the hand model could be achieved quite elegantly through constrained gradients. We have extended the basic SMD scheme for the purpose of tracking in several ways. As tracking is in fact a series of optimizations for subsequent frames, it was ensured that these would not start from scratch, but would gain robustness and speed by taking over a good initialization for the step sizes from the previous frame. Then, it has been shown that including the surface orientation information in the cost function increases its robustness.

Our experiments show that the SMD tracker performs robustly and efficiently on rather complex 3D hand sequences. Performance came out to be better than that of several, alternative methods running on the same data. The proposed tracker is sufficiently generic to be adapted to different types of input. If the cost function is adapted in an appropriate way, it can use 2D instead of 3D features as well. A natural extension of this work would be to include 2D features in order to disambiguate complex situations. Another ongoing improvement is aimed at further increases in robustness. While the stochastic sampling approach helps to overcome local minima, it is not guaranteed that the global optimum will be found. Combining several SMD trackers as 'smart particles' in a Condensation framework is one of the avenues that we are currently exploring where the first results have been presented in [8].

References

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

[2] V. Athitsos, S. Sclaroff, An appearance-based framework for 3D hand shape classification and camera viewpoint estimation, International Conference on Automatic Face and Gesture Recognition, 2002, pp. 40–45.

[3] D. Bertsekas, Nonlinear Programming, Athena Scientific, 2000.

[4] P.J. Besl, H.D. McKay, A method for registration of 3D shapes, IEEE Transactions on Pattern Analysis and Machine Intelligence 14 (1992) 239–256.

[5] M.J. Black, A. Jepson, Eigentracking: robust matching and tracking of articulated objects using a view-based representation, European Conference on Computer Vision, 1996, pp. 329–342.

[6] M. Brand, Shadow puppetry, International Conference on Computer Vision, 1999, pp. 1237–1244.

[7] M. Bray, E. Koller-Meier, P. Müller, L. Van Gool, N.N. Schraudolph, 3D hand tracking by rapid stochastic gradient descent using a skinning model, First European Conference on Visual Media Production (CVMP), 2004, pp. 59–68.

[8] M. Bray, E. Koller-Meier, L. Van Gool, Smart particle filtering for 3D hand tracking, International Conference on Automatic Face and Gesture Recognition, 2004, pp. 675–680.

[9] E.E. Catmull, A system for computer generated movies, ACM Annual Conference, 1972, pp. 422–431.

[10] T.J. Cham, J. Rehg, A multiple hypotheses approach to figure tracking, International Conference on Computer Vision and Pattern Recognition, 1999, pp. 239–245 (II).

[11] Q. Delamarre, O.D. Faugeras, 3D articulated models and multiview tracking with physical forces, Computer Vision and Image Understanding 81 (2001) 328–357.

[12] J. Deutscher, A. Blake, I. Reid, Articulated body motion capture by annealed particle filtering, International Conference on Computer Vision and Pattern Recognition, 2000, pp. 26–133.

[13] J.-P. Gourret, N.M. Thalmann, D. Thalmann, Simulation of object and human skin formations in grasping task, Proceedings of the 16th Annual Conference on Computer Graphics and Interactive Techniques, 1989, pp. 21–30.

[14] A.J. Heap, D.C. Hogg, Towards 3D hand tracking using a deformable model, International Conference on Automatic Face and Gesture Recognition, 1996, pp. 140–145.

[15] M. Isard, A. Blake, Condensation—conditional density propagation for visual tracking, International Journal on Computer Vision 29 (1998) 5–28.

[16] R.A. Jacobs, Increased rates of convergence through learning rate adaptation, Neural Networks 1 (1988) 295–307.

[17] J. Kivinen, M.K. Warmuth, Additive versus exponentiated gradient updates for linear prediction, Proceedings of the 27th Annual ACM Symposium on Theory of Computing, 1995, pp. 209–218.

[18] K. Komatsu, Human skin model capable of natural shape variation, The Visual Computer 4 (3) (1988) 265–271.

[19] T.P. Koninckx, A. Griesser, L. Van Gool, Real-time range scanning of deformable surfaces by adaptively coded structured light, 3D Digital Imaging and Modeling (2003) 293–300.

[20] J.J. Kuch, T.S. Huang, Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration, International Conference on Computer Vision, 1995, pp. 666–671.

[21] K. Levenberg, A method for the solution of certain non-linear problems in least squares, Quarterly Journal of Applied Mathematics II (2) (1944) 164–168.

[23] J. Lin, Y. Wu, T.S. Huang, Modeling human hand constraints, ARL Federated Laboratory Fifth Annual Symposium, 2001, pp. 105–110.

[24] M.H. Lin, Tracking articulated objects in real-time range image sequences, International Conference on Computer Vision, 1999, pp. 648–653.

[25] N. Magnenat-Thalmann, R. Laperrire, D. Thalman, Jointdependent local deformations for hand animation and object grasping, Graphics Interface (1988) 26–33.

[26] D.W. Marquardt, An algorithm for least-squares estimation of non-linear parameters, Journal of the Society of Industrial and Applied Mathematics 11 (2) (1963) 431–441.

[27] R. Plaenkers, P. Fua, Articulated soft objects for video-based body modeling, International Conference on Computer Vision, 2001, pp. 394–401.

[28] W.H. Press, B.P. Flannery, S.A. Teukolsky, W.T. Vetterling, Numerical Recipes in C, Cambridge University Press, Cambridge, 1988.

[29] M. Proesman, L. Van Gool, A. Oosterlinck, One-shot active range acquisition, International Conference on Pattern Recognition, 1996, pp. 336–340.

[30] J.M. Rehg, T. Kanade, Visual tracking of high DOF articulated structures: an application to human hand tracking, European Conference on Computer Vision, 1994, pp. 35–46.

[31] J.M. Rehg, T. Kanade, Model-based tracking of self-occluding articulated objects, International Conference on Computer Vision, 1995, pp. 612–617.

[32] B.A. Pearlmutter, Fast exact multiplication by the hessian, Neural Computation 6 (1) (1994) 147–160.

[33] R. Rosales, V. Athitsos, L. Sigal, S. Sclaroff, 3D hand pose reconstruction using specialized mappings, International Conference on Computer Vision, 2001, pp. 378–385.

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

[35] N. Shimada, K. Kimura, Y. Shirai, Real-time 3D hand posture estimation based on 2D appearance retrieval using monocular camera, Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems, 2001, pp. 23–30.

[36] H. Sidenbladh, M.J. Black, D.J. Fleet, Stochastic tracking of 3D human figures using 2D image motion, European Conference on Computer Vision, 2000, pp. 702–718.

[37] C. Sminchisescu, B. Triggs, Covariance scaled sampling for monocular 3D body tracking, International Conference on Computer Vision and Pattern Recognition, 2001, pp. 447–454.

[38] R.S. Sutton, Adapting bias by gradient descent: an incremental version of delta-bar-delta, Proceedings of the 10th National Conference on Artificial Intelligence, 1992, pp. 171–176.

[39] C. Tomasi, S. Petrov, A. Sastry, 3D tracking = classification + interpolation, International Conference on Computer Vision, 2003, pp. 1441–1448.

[40] Y. Wu, J. Lin, T.S. Huang, Capturing natural hand articulation, International Conference on Computer Vision, 2001, pp. 426–432.