

# Tracking Of Hand's Posture And Gesture

F. Le Jeune  
R. Deriche  
R. Keriven  
P. Fua

Research Report 04-02  
October 2004



**CERTIS, ENPC,**  
77455 Marne la Vallée, France,  
<http://www.enpc.fr/certis/>



# Tracking Of Hand's Posture And Gesture

## Suivi tridimensionnel de la main

F. Le Jeune<sup>2</sup>  
R. Deriche<sup>2</sup>  
R. Keriven<sup>2</sup>  
P. Fua<sup>1</sup>

---

<sup>1</sup>Cvlab, EPFL, Swizerland, <http://cvlab.epfl.ch/>

<sup>2</sup>Odyssee Project, INRIA/ENPC/ENS, France, <http://www-sop.inria.fr/odyssee/>



## **Abstract**

The hand with its pose and gesture is not easy to track with cameras. It is mostly because of a high number of degrees of freedom and a complex shape. Our algorithm is based on three dimensional clouds of points issued from stereovision in order to keep the depth information. The complex and organic aspect of the hand is modeled by an articulated skeleton on which metaballs are attached. We reduce the difficulty of the problem by adding joint constraints to the model. The core of our program is an energy-based minimization that fits the model at each frame using the distance between the model and the clouds of points, the result at the previous frame is chosen as an initial guess.

## **Keywords**

Computer vision, hand, 3D model, constraints



## Résumé

La position et la configuration d'une main en mouvement ne sont pas faciles à retrouver pour un ordinateur équipé de caméras. Cela est en partie dû au nombre élevé de degrés de liberté de cet objet ainsi que de sa forme complexe. Nous avons basé notre méthode de suivi sur des nuages de points en trois dimensions obtenus par stéréovision afin de conserver l'information de profondeur. L'aspect complexe et organique de la main a été modélisé par un squelette articulé habillé de metaballs. Quant à la difficulté liée au grand nombre de degrés de liberté, nous nous sommes efforcés de la réduire en rajoutant des contraintes articulaires. La recherche de la position correcte à partir de la position précédente estimée repose sur la minimisation d'une énergie basée sur la distance entre le modèle de la main et le nuage de points en trois dimensions.

### Mots Clef

Vision par ordinateur, suivi de la main, modèle articulé, contraintes





# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
<b>2</b>	<b>Data Acquisition</b>	<b>3</b>
2.1	Camera Calibration . . . . .	3
2.2	Color Reconstruction From Bayer Image . . . . .	5
2.3	Filtering the Points Composing the Cloud . . . . .	8
<b>3</b>	<b>Complete Hand Model</b>	<b>9</b>
3.1	Articulated Skeleton . . . . .	9
3.2	Ellipsoids and Metaballs . . . . .	9
3.3	Distance to an Ellipsoid . . . . .	11
3.4	Constraints . . . . .	13
3.5	Simplified Hand Models . . . . .	14
<b>4</b>	<b>Searching for the best fit</b>	<b>15</b>
<b>5</b>	<b>Experimental Results</b>	<b>17</b>
5.1	Display of the results . . . . .	17
5.2	Tests on Sequences . . . . .	17
<b>6</b>	<b>Conclusion</b>	<b>19</b>



## 1 Introduction

Tracking human hand gestures has been an important subject of research for years. All existing systems have their advantages and their drawbacks. Systems involving gloves with sensors were the first ones to give results and they are still the technology that provides best data and precision. But the problem is that such solutions, even the wireless ones, are invasive because gloves are not as thin as silk gloves and need to carry sensors.

Systems involving instrumented gloves such as the one shown in Fig.1 were the first to give useful results and are still the technology that provides the most reliable data. However such gloves, including the wireless ones, are cumbersome because they are not as thin as silk gloves and need to carry sensors. By contrast, filming a subject using one or more cameras does not disturb him. As a result, research teams trying to capture the most natural gesture possible, focus on video-based solutions.



Figure 1: Nathaniel Lynch (Microsoft) wearing an invasive acquisition glove.

There are many different approaches to track human movements from video cameras; in his survey [13], the author enumerated 130 of them and more have been proposed since. In most cases, the hand is first roughly located by finding a particular color in the image, be it skin color or the color of gloves worn by the subject. Various techniques are then used to recognize the hand position and/or gesture. The oldest ones are those using one camera and comparing features of the picture to a set of pictures of the hand. For example, W. T. Freeman controlled a TV set using 26 predefined hand gestures [14] and N. Shimada [17] proposed something similar. This approach works only when the set of predefined gestures is not too big and these gestures differ sharply from one another. Another class of methods is formed by those that track the movement of the hand to recognize dynamic

gestures. The detected hand motion is usually sent to a Hidden Markov Model procedure. T. Starner uses this to recognize sentences in sign language composed of roughly 40 signs [15]. This technique is quite efficient but limited since it cannot detect the difference between two gestures involving identical global motion but a different hand shape. It is limited also by the fact that using only one camera does not allow to easily measure motion in the depth plane. Finally, the most complex class of methods includes those that rely on 3-dimensional analysis. Most of them fit features of the projection of a 3-dimensional articulated model to image features, which tends to be easier when using multiple-views and fitting 3D models to 3D data. The most commonly used features are contours [2, 19, 18, 20, 21], shading [16] and stereo [2, 4] as shown in Fig. 2.

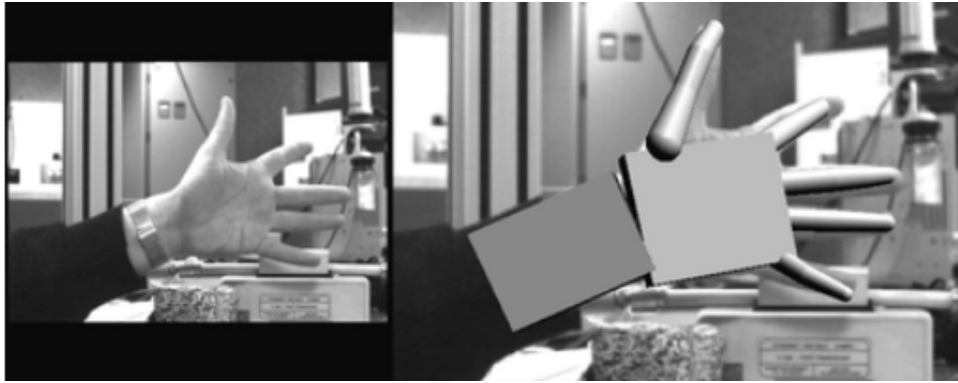


Figure 2: Quentin Delamarre (INRIA).

The approach proposed here builds on these earlier ones. We use a multi-camera approach to remove the ambiguities inherent to mono-camera systems due to the lack of depth perception. For each frame, we obtain a 3-dimensional cloud of points from stereovision and try to fit an articulated hand model to it. We chose the articulated model solution instead of the database one because we wanted to be able to recognize and follow every kind of hand position without having to learn each possible hand position.

Our goal is to track the hand by fitting a full 3-dimensional model to incomplete and imperfect input data. The main problems we had to face stem from the noisyness of our 3D data, occlusion and the complexity of our articulated model with its 26 degrees of freedom. To accomplish this task we defined a hierarchy of models that yield satisfactory tracking results by simply minimizing in the least-squares sense the distance of our model to the data.

## 2 Data Acquisition

At the beginning of this project, we obtained our clouds of points from pre-calibrated cameras made for stereovision: the *Digiclops* from the *Pointgrey* company and their own API [25]. The advantage of this system is that it directly outputs a cloud of points out of the box. It was simple and easy but it lacked of control of adjustment and of the possibility to use our own algorithms and filters. One of the biggest problems was that those cameras have a fixed focus; fixed to clearly see a man several meters away but not to clearly see a close hand. We therefore switched to our own stereovision algorithm. We first tested it with synthetic sequences and known virtual cameras calibration parameters and, then, with real cameras that we calibrated ourselves.

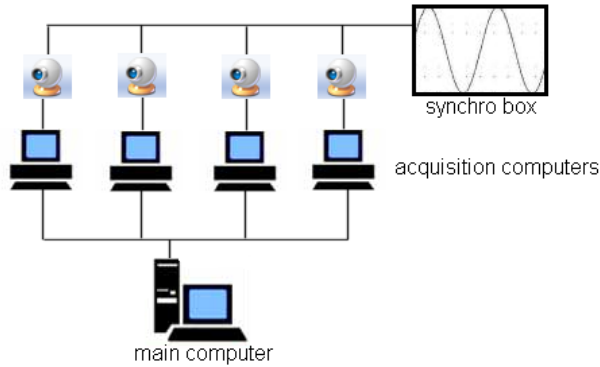


Figure 3: network used to do synchronized multi-camera acquisition.

The acquisition system we use is based on several digital color cameras, as shown in Fig. 3. They are synchronized together using an external signal in order to grab frames at the same instant  $t$ . In order to be less limited by computer bandwidth, we chose to assign one computer to each camera, to store frames in the memory of the computer during the acquisition and to save all the pictures taken on a main computer when the sequence ends. To remove one acquisition slowdown, we decided to grab raw data in real time and then to process them to get the clouds of points offline.

### 2.1 Camera Calibration

Our data processing line contains several steps. The first one is the calibration step. We use Intel OpenCV library [23] to detect the inner corners of a moving grid shown in Fig. 4 in a 100 frames sequence and to calibrate the cameras from the detected points.



Figure 4: grid used for calibration.

For each camera, the calibration estimates those values:

- 5 parameters to build the intrinsic transformation matrix:

$$KK = \begin{bmatrix} fc(1) & \alpha_c * fc(1) & cc(1) & 0 \\ 0 & fc(2) & cc(2) & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

- 12 parameters defining the rotation and the translation matrix:

$$R_c = \begin{bmatrix} R_1 & R_2 & R_3 & 0 \\ R_4 & R_5 & R_6 & 0 \\ R_7 & R_8 & R_9 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$T_c = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Using those matrix, the projection of a 3D point  $X_{3D} = \begin{pmatrix} x_{3D} \\ y_{3D} \\ z_{3D} \\ 1 \end{pmatrix}$

to a 2D point of the screen  $X_{2D} = \begin{pmatrix} x_{2D} \\ y_{2D} \\ k_{2D} \end{pmatrix}$  can be done using  $X_{2D} = KK.R_c.T_c.X_{3D}$ . Without distortion, 2D coordinates on screen are  $\left(\frac{x_{2D}}{k_{2D}}, \frac{y_{2D}}{k_{2D}}\right)$ .

- 5 distortion parameters  $k_c(1..5)$  that are used to compute the distorted coordinates. Let  $X = \begin{pmatrix} x \\ y \end{pmatrix}$  be the projected coordinates and  $X_d = \begin{pmatrix} x_d \\ y_d \end{pmatrix}$  the distorted coordinates, then:

$$X_d = (1 + k_c(1)r^2 + k_c(2)r^4 + k_c(5)r^6) X + d_x$$

with

$$r^2 = x^2 + y^2$$

$$d_x = \begin{bmatrix} 2k_c(3)xy + k_c(4)(r^2 + 2x^2) \\ k_c(3)(r^2 + 2y^2) + 2k_c(4)xy \end{bmatrix}$$

The transformation from 3D coordinates to 2D coordinates is used by the algorithm that creates the cloud of 3D points from a pair of 2D calibrated images.

We use a correlation-based algorithm to compute disparity maps for image-pairs [5, 11] and, when more than two cameras are used, we merge the corresponding point clouds to create a denser result [11].

## 2.2 Color Reconstruction From Bayer Image

To keep most information from the cameras and to have the highest frame-rate, we directly acquire raw images and rectify them later to color and greyscale images. The problem was that in our raw images the color is coded using a Bayer pattern, which means a 66% loss of information if compared to an image where the red, green and blue components are known for each pixel. We chose not to use neither hardware Bayer reconstruction filter nor simple formulae to compute the color of each pixel. The reasons are that the hardware filter is slow and prevent realtime data acquisition and that the software filters either draw some regular saw patterns that may be stronger than the texture along horizontal or vertical lines or output some blurred or color distorted images. We preferred to create our own anisotropic color reconstruction filter which produces sharp images with less visible Bayer pattern.

The Bayer pattern is a technique used to have color pictures from greyscale sensors. It consists in putting a red, green or blue transparent filter in front of each pixel of the greyscale sensors (this is done by the camera manufacturer, not by the user). The advantage of this technique is that a color camera cost roughly the same price as a greyscale camera. But the drawback are the missing color pixels that needs to be reconstructed. Since

our eyes are more sensitives to the green component, the Bayer pattern used twice more green pixels than red or blue one. A typical Bayer mosaic looks like this :

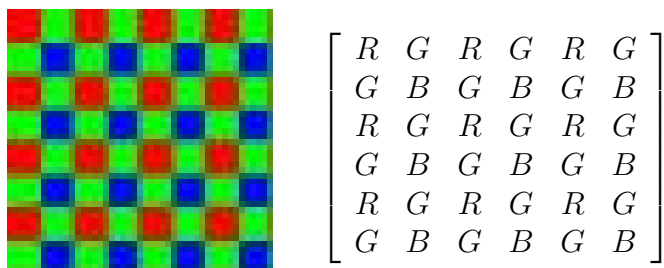


Figure 5: Typical Bayer mosaic.

which can be seen as:

$$\begin{bmatrix} R & . & R & . & R & . \\ . & . & . & . & . & . \\ R & . & R & . & R & . \\ . & . & . & . & . & . \\ R & . & R & . & R & . \\ . & . & . & . & . & . \end{bmatrix} + \begin{bmatrix} . & . & . & . & . & . \\ . & B & . & B & . & B \\ . & . & . & . & . & . \\ . & B & . & B & . & B \\ . & . & . & . & . & . \\ . & B & . & B & . & B \end{bmatrix} + \begin{bmatrix} . & G & . & G & . & G \\ G & . & G & . & G & . \\ . & G & . & G & . & G \\ G & . & G & . & G & . \\ . & G & . & G & . & G \\ G & . & G & . & G & . \end{bmatrix}$$

Two third of the values are missing. The easy way to compute the color of a pixel is to compute for missing component a linear interpolation of all the known values in a 8-neighborhood. It is simple but creates some artefact in the green channel in case of strong horizontal or vertical gradient (because of the diagonal alignment of green sensors) as seen in Fig. 6. And this appears strongly in the final image because the green channel has the heighest weight when converting a color image to a greyscale one using:

$$Grey = 0.299 \times Red + 0.587 \times Green + 0.114 \times Blue$$

This artefact can be bigger than the texture and can create correlation errors since the matching algorithm may prefer to align two artefacts instead of two real textures. To remove it while keeping sharp images we replace the linear interpolation of the green component by a more complex interpolation.

To compute intensity at point  $P$  we first estimate the direction of the isolevel line from the computing of finite differentiate of the four neighbours

$$\begin{bmatrix} P_1 & & P_2 \\ & P & \\ P_3 & & P_4 \end{bmatrix} :$$

$$\Delta_x = \frac{(P_2 - P_1) + (P_4 - P_3)}{2}$$



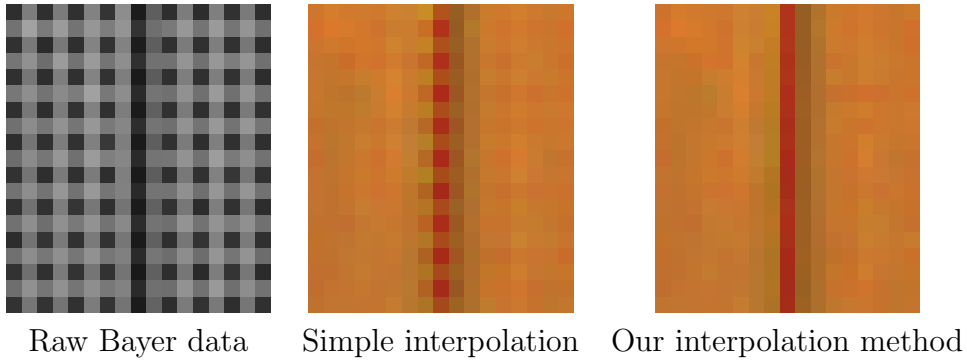


Figure 6: Reconstructing of a color image from Bayer mosaic using different interpolation algorithms

$$\Delta_y = \frac{(P_3 - P_1) + (P_4 - P_2)}{2}$$

The orientation of the isometric line is  $(-\Delta_y, \Delta_x)$ . This line cuts the square made of the four pixels in two points ( $P'$  and  $P''$ ). The intensity value of those points is estimated as a barycentric mean of the value of each extremity of the segment. The intensity value of point  $P$  is a barycentric mean of the two intersections points. In Fig. 7,  $P'$  is a mean between  $P_1$  and  $P_2$ ,  $P''$  is a mean between  $P_2$  and  $P_4$  and the value of  $P$  is a mean between  $P'$  and  $P''$ ; in this example the value of  $P$  should be close to  $P_2$ .

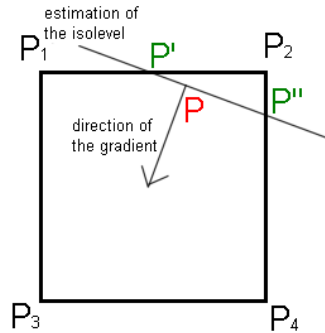


Figure 7: Interpolation of intensity value.

Two examples with drawing of the estimation of the isometric line are shown in Fig. 8.

Using this technique gives better results than a simpler of Fig. 6 and since it can give intensity value for pixels located on non discrete coordinates, we use it. This is usefull since the rectification algorithm and the distortion

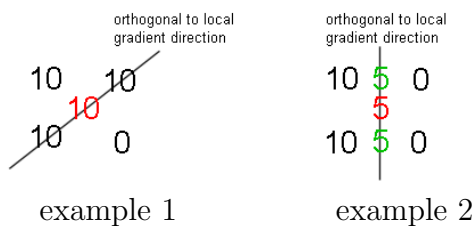


Figure 8: Interpolation of central value using local gradient

formulae needs pixel color at real coordinates.

Then we compute disparity on greyscale rectified images and filter them [12, 5]. We keep a trace of the color of each pixel for filtering purpose. Using this method, we obtain good quality clouds, due to both the good resolution of the input pictures, at least  $700 \times 500$ , and the quality of the Bayer reconstruction.

## 2.3 Filtering the Points Composing the Cloud

Then to clean the cloud of points, we used several filters.

The use of the position of the hand on the previous frame allows a first filtering of the cloud by removing every point that is too far from a probable current position. In practice, points that are more far than a distance  $d_{max}$  from the hand position at the previous frame are removed.  $d_{max}$  is set by the user and even with a big value (around ten centimeters), it works and removes a lot of points from the background. Most of remaining points belong to the hand.

Another filtering technique [10] is used to remove the last useless points: a color filter. The  $(R, G, B)$  color is projected to a two dimensional space  $(r, g)$  using:

$$\begin{cases} r = \frac{R}{R+G+B} \\ g = \frac{G}{R+G+B} \end{cases}$$

The zone corresponding to the color of the skin is the inner of an ellipse, as shown in Fig. 9. The parameters of this ellipse (global size, ratio between axis and general orientation) are computed from color histograms get from manually segmented images taken under similar lighting condition (two images are enough). The threshold of the filter can be set by changing the global size of the ellipse.

Those two filters let us remove most of the points that does not correspond to the hand. They are not able to correct the error in depth that we had mostly using the *Digiclops* cameras. The depth error was solved using better

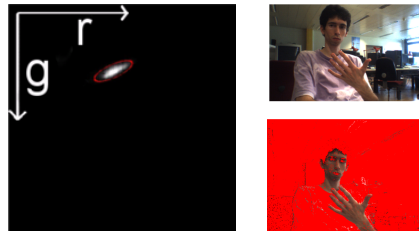


Figure 9: zone corresponding to skin color in  $(r, g)$  space and application of this filter on an image.

cameras.

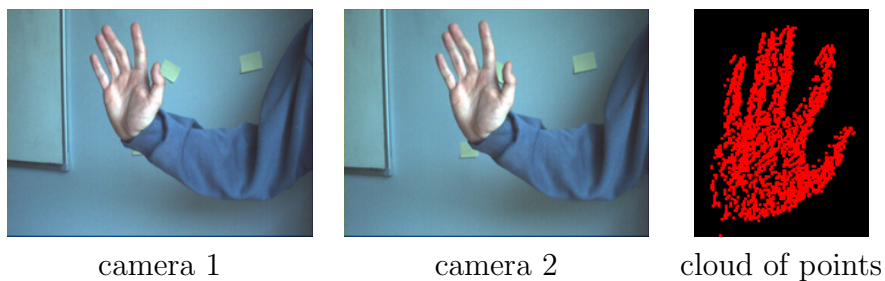


Figure 10: filtered cloud obtained from two calibrated pictures.

## 3 Complete Hand Model

### 3.1 Articulated Skeleton

We based the creation of our articulated hand model on the work of Q. Delamarre et O. Faugeras[2] and J. Rehg et T. Kanade[3]. When no constraint is applied this model has 26 degrees of freedom. The validation of the model has been done using the creation of an application that shows the hand model displaying each 26 lettres of the French Sign Langage, as shown in Fig. 11.

### 3.2 Ellipsoids and Metaballs

The volume of the hand is given by metaballs: they are fixed to the bones of the skeleton and merge together to form the fingers and the palm. A total of 16 ellipsoids are used: one for the palm and three for each finger. It is possible to attach more ellipsoids to the skeleton in order to have a better model.

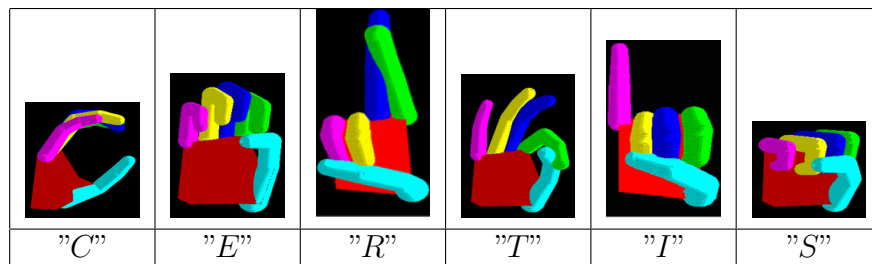


Figure 11: examples of positions taken by our model

The surface of the hand is seen as an isosurface of the sum of potential functions [7]. We chose the distance to the ellipsoid as the potential function and the isosurface where the value of the function is 1. The global potential function can be written as

$$d(X, \alpha) = -\log \left( \sum_{e=1}^{nbEllipsoids} \exp^{-dist_e(X, \alpha)} \right)$$

where  $d(X, \alpha)$  is the signed distance between the point  $X$  and the surface of the hand whose position parameters are defined by the vector  $\alpha$ .  $dist_e(X, \alpha)$  is the signed distance between the point  $X$  and the surface of the ellipse  $e$  from the hand model.

Using metaballs, and by this way estimating the distance to the model by a non trivial function, creates a model which has a smoother, more realistic and more organical aspect than if we decided to estimate the distance to the model by the distance to the closest ellipsoid, as shown in Fig. 12.

To prevent fingers to fusion, which is a classic artefact when merging metaballs, the distance function  $d$  is actually computed as the sum of the distance to some selected ellipsoids[1]. The selection of the ellipsoids used depends on the position of the point  $X$ : this is the closest ellipsoid and every ellipsoid that is directly linked to this one by a bone of the skeleton.

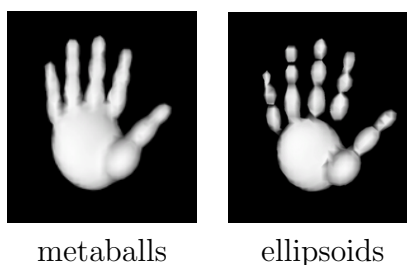


Figure 12: surface of the model depending of chosen distance estimation

### 3.3 Distance to an Ellipsoid

Computing the distance between a point and the surface of an ellipsoid is not simple. This value has to be estimated. The main difficulty is that although the intersection between the segment linking the point  $P$  and the center of the ellipsoid is easily computable ( $I$  on the scheme 13), this is not the point on the surface that is the closest to the point  $P$ . The point on the surface that minimizes this distance is the point named  $D$  on the scheme. The coordinates of this point are not computable using simple usual functions.

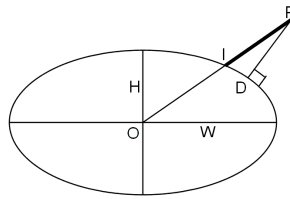


Figure 13: distance  $PD$  approximated by distance  $PI$ .

A first simplification consists in a deformation of the object  $\{\text{point } P, \text{ellipsoid}\}$  to transform the ellipsoid into a sphere and the point  $P$  into a point  $P'$ . Then the distance between the point  $P'$  and the sphere is easily computable and can be used as a first approximation of the distance between the point  $P$  and the surface of the ellipsoid. This distance is named  $d_1(x, y)$  on Fig. 14.

This estimation of the distance decreases faster in the direction of the smallest axe of the ellipsoid than in the direction of the biggest. In order to have a constant decrease of the estimation of distance at least along the three axis, we use another distance estimation of  $PI$ , as suggested in [1]. To simplifiat the formulae, here is the case of a two-dimensional ellipse, centered on  $O$  and whose axis are aligned along  $Ox$  and  $Oy$ .

Let  $M = \begin{bmatrix} \frac{1}{W} & 0 \\ 0 & \frac{1}{H} \end{bmatrix}$  the matrix that transforms the ellipse  $(W, H)$  into a circle of radius 1. The point  $I$  is the intersection of  $(OP)$  and the ellipse. Let  $I'$  and  $P'$  the images of  $I$  and  $P$  after applying this transformation. The image of  $O$  is still  $O$ . We obtain that  $I'$  is the intersection between the line  $(OP')$  and the unitarian circle.

$$I' = \frac{1}{|P'|} \begin{pmatrix} \frac{x}{W} \\ \frac{y}{H} \end{pmatrix}$$

Coordinates of  $I$  can then be computed.

$$I = M^{-1} \cdot I' = \frac{1}{|P'|} \begin{pmatrix} x \\ y \end{pmatrix}$$

The signed distance  $IP$  can be obtained easily and the final result is:

$$d_2(x, y) = \sqrt{x^2 + y^2} \cdot \left( 1 - \frac{1}{\sqrt{\left(\frac{x}{W}\right)^2 + \left(\frac{y}{H}\right)^2}} \right)$$

Even if this distance is still an approximation, it is a good compromise between a near perfect estimation of the distance to an ellipsoid and an easy but totally incorrect estimation. This function is differentiable and we use those differentiates in the minimization algorithm.

The good compromise of the distance is especially true outside the ellipsoid. In the inner region,  $d_2$  has some isometric lines quite different from those of the real distance (the real distance has lines in forme of '0',  $d_2$  has lines in form of '8'). We decided to approximate the ellipsoid by a sphere when computing the inner distance, without applying the correction we used outside to have better values along the axis. We just use a simple affine transformation in order that the approximation of the signed distance is 0 near the surface and  $-r$  at the center (with  $r$  = the half-length of the smallest of the three axis).

Finally we use:

$$d_3(x, y) = \begin{cases} \left( \sqrt{\left(\frac{x}{W}\right)^2 + \left(\frac{y}{H}\right)^2} - 1 \right) \cdot r & \text{inside} \\ d_2(x, y) & \text{outside} \end{cases}$$

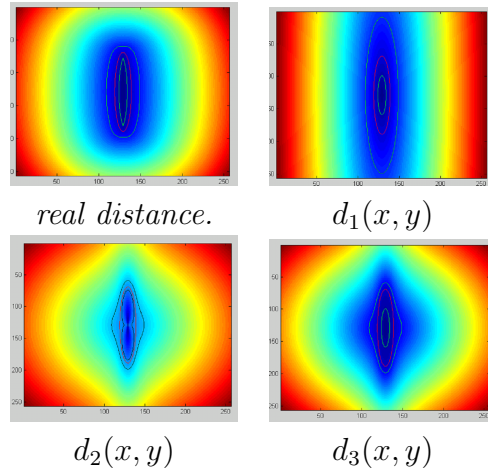


Figure 14: approximated distances.

Even if we use an approximation of the distance to the surface of the ellipsoid, the final result is satisfying as shown in Fig. 14. Moreover, the fact

of using a function differentiable on most points of the space to approximate this distance allow us to obtain analytically for each point of the space its distance to the surface of the hand and the differentiates of this distance with respect to each parameter of the model. As in [4], this result allows us to use the Levenberg-Marquardt [22] minimization algorithm in order to find the best parameters that minimize the distance between the cloud of points and our hand model.

### 3.4 Constraints

In order to force our algorithm to search solutions in a space of realistic positions of the hand, we add some constraints to the articulations of the model.

The first kind of constraints is of type *min/max* and set limits to each parameter independently of each other. A each step of the Levenberg-Marquardt algorithm, the values of each parameter is tested and if it is out its restricted interval, it is reprojected to the closest valid value. Those constraints are simple to define and to test and allow a great increase of the goodness of the output of the algorithm.

The second kind of constraints are here to set the value of one parameter from the value of another parameter. They are essentially used for the last articulation of each finger where the value of bending is set as two third of the value of the articulation just above, as shown in Fig. 15. This constraint that can be found in [2] is commonly used in the world of hand tracking methods using articulated model. This allows to reduce the number of degrees of freedom from 26 to 21.

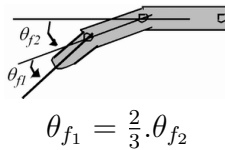


Figure 15: constraint on the last articulation.

We tried to add the same kind of constraint on the abduction<sup>3</sup> values of the four fingers. The abduction values are linked together using this equation:

$$\begin{bmatrix} c_0 & c_1 & c_2 & c_3 \end{bmatrix}^t \cdot \alpha = \begin{bmatrix} A_0 & A_1 & A_2 & A_3 \end{bmatrix}$$

where  $A_0, A_1, A_2, A_3$  are the abduction values of the index finger, the middle finger, the ring finger and the little finger and  $c_0, c_1, c_2$  and  $c_3$  are constants

<sup>3</sup>the abduction is the ability the spread the fingers.

that are peculiar for each person and needs to be measured before the tracking. This constraint allows to replace four abduction values by an unique parameter  $\alpha$ . This doesn't seem to be used yet by any other research team. The first tests we did don't show clearly if this constraint is valid or not. Sometimes it is, sometimes it isn't. It seems that it works only when the dimensions and lengths of the model exactly correspond to the acquired hand. This is not always the case until we work on a good system of automatic model calibration.

When used, this constraint reduces the number of parameters of the model by 3 and sets its final number to 18 (6 degrees for the global position in space and 3 parameters for the thumb and 9 parameters for the four other fingers).

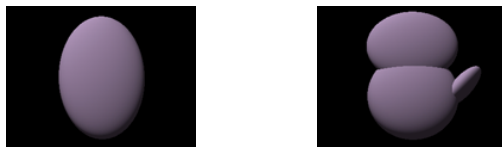
A third kind of constraint would be some *min/max* one whose boundaries *min* and *max* are not fixed but depends on the value of other parameters. It could be used to model the fact that the abduction interval depends on the bending of the finger (there is not abduction possible when the finger is bended with a value higher than 90 degrees)

### 3.5 Simplified Hand Models

When there is not enough data to fit the whole model of the hand because of noise, illumination problem or a big occlusion, we can use the simplified models of Fig. 16:

1. The simplest is made of one single ellipsoid. It has six degrees of freedom, 3 translation and 3 rotation, and those six parameters can be estimated even when the cloud of points is not precise enough for a 26 degrees of freedom model. In use, it appears that the three parameters corresponding to the center of the ellipsoid are found but not always the three one corresponding to the orientation of the hand. This liberty of orientation taken by the model is due to the poor similarity between this model and a real hand.
2. We solve this problem by using a slightly more complex model of the hand depicted by Fig. 16. Composed of three ellipsoids and looking like a glove, this model is simple enough to have a reduced number of degrees of freedom (9 degrees) and to have big enough ellipsoids in order not to be influenced by zone of lack of data in the cloud of points, which usually correspond to some fingers that the reconstruction algorithm is not able to reconstruct. We performed some successful tests with this model and they show that it can track the hand on sequences where the complex model is lost by the poor quality of the clouds of points.





6 degrees of freedom    9 degrees of freedom

Figure 16: the two simple models (drawn without metaballs)

The calibration of the dimensions and lengths of the model is currently manually done. Our long term objective is to do it automatically from pictures of the acquired hand.

## 4 Searching for the best fit

The tracking of the hand is currently done frame by frame, the initial position used by the algorithm at frame  $n$  is the final position found at frame  $n - 1$ . Currently the initial position of the sequence is manually set. The algorithm used to minimize the cost function of the distance between the model and the cloud of points is the Levenberg-Marquardt algorithm[22]. It can minimize functions that can be written this form:

$$F(\alpha) = \sum_{i=1}^n \left( \frac{f(i, \alpha) - y_i}{\sigma_i} \right)^2$$

Our first idea was to set  $f(i, \alpha) = d_3(X_i, \alpha)$  where  $i$  travels through the  $n$  points of the cloud,  $y_i = 0$ ,  $\sigma_i = 1$  and  $\alpha$  is the vector containing the parameters of the model. The use of this tracking method is the result of a collaboration between Odyssee (INRIA/ENPC/ENS) and CVlab (EPFL) and extends the work of [9].  $F(\alpha)$  is seen as an energy to minimize.

$$E_1(\alpha) = \sum_{i=1}^n \left( \frac{d_3(X_i, \alpha)}{1} \right)^2$$

$$F_1(\alpha) = E_1(\alpha)$$

We decided to modify this energy to prevent some parts of the model to stay in empty zones. We discretize the surface of the model using the Marching Cubes algorithm [6] (it is done at the beginning of the minimization, then each point is fixed to its closest bone), we precompute a discrete approximation of the distance of any voxel in a zone near the hand to the

cloud of points using the Danielsson's algorithm[8], we add this as a second term of the energy to minimize and finally write it this form:

$$E_2(\alpha) = \sum_{j=1}^{n_2} \left( \frac{f_2(j, \alpha) - y_j}{\sigma_j} \right)^2$$

$$F_2(\alpha) = K E_1(\alpha) + (1 - K) E_2(\alpha)$$

Where  $n_2$  is the number of points of the discretized model,  $y_j = 0$ ,  $\sigma_j = 1$ ,  $f_2(j, \alpha)$  is an estimation of the distance between the point  $j$  of the model and the closest point of the cloud and  $K$  is a value set by the user to balance both energy. As shown in Fig. 17,  $E_1$  and  $E_2$  are both needed to do a good fitting. If only one of those energy is used, it may happen cases where energy is low but fitting is incorrect as seen in Fig. 17.



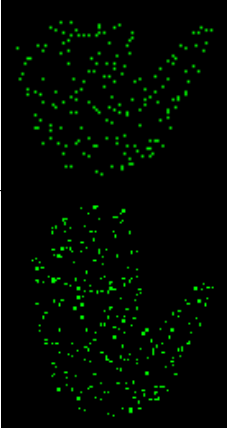
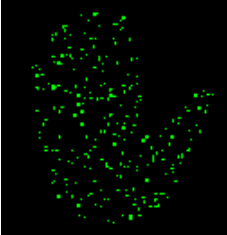
Position of model → ↓ cloud of points		
	$E_1$ : LOW $E_2$ : LOW	$E_1$ : <b>LOW</b> (but <b>wrong</b> fitting!) $E_2$ : HIGH
	$E_1$ : HIGH $E_2$ : <b>LOW</b> (but <b>wrong</b> fitting!)	$E_1$ : LOW $E_2$ : LOW

Figure 17:  $E_1$  and  $E_2$  corresponding to particular configurations of hand and cloud.

Sometimes the algorithm doesn't find the right solution. One of the reason why the algorithm does not give the expected result is that it stucked in a local minimum. A situation of local minimum that we face frequently is when one finger of the model is fitted with a different finger in the cloud of point. A solution to go out of this minimum is to relaunch a serie of minimization from two new positions obtained from the current solution. Fig. 18 shows

one with all fingers shifted in one direction, one with all fingers shifted in the other direction.



Figure 18: local minima.

This method prove to give good results but can unintentionally push the tracking towards a bad position if the cloud of point is not good enough. In practice this solve more problems than it gives errors.

## 5 Experimental Results

### 5.1 Display of the results

To display the surface of our model defined as an implicit distance function, we use the Marching Cubes algorithm [6]. We discretize the space around the hand into a regular grid then we compute the value of the implicit distance function at each node of the grid and finally we use the Marching Cubes algorithm to compute the surface and display it using a simple OpenGL[24] based viewer

The main drawback of this method is the computing time of distance function at each node of the grid. It cannot be used in a software that needs to display the model in real time.

A faster solution consists in drawing directly the ellipsoids used by the metaballs. The result is less nice, smooth and realistic but is enough to show the position of the hand. This fast method is used when manually setting the initial parameters of the hand for first frame and when displaying the tracking animation in real time.

The display based on the Marching Cubes algorithm is used to visualize the result of the tracking with precision. What is shown is exactly what the minimization algorithm sees.

### 5.2 Tests on Sequences

Our algorithm was tested on three kinds of sequences.

The first one, depicted by Fig. 19, are real sequences taken by the *Digiclops* at a  $320 \times 240$  resolution that are not so good because the *Digiclops*



Figure 19: images extracted from the *Digiclops*.

seems to have been used in less than ideal conditions. The tracking of the hand using the full articulated model is a failure. The main cause is the cloud of points that is far to represent the reality (some zone of the hand doesn't have any 3D points and some points have a depth that is more than several centimeters wrong). The tracking of the movement with a simplified hand model which has 9 degrees of freedom and four fingers gathered in one ellipsoid, as shown in Fig. 16 gives better results.

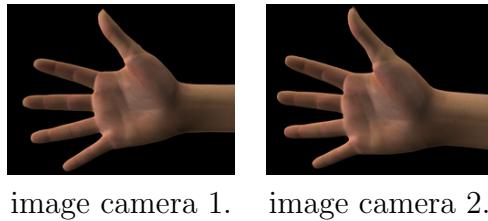


Figure 20: images extracted from the synthesis sequence.

The other kind of sequence is a synthetic sequence whose resolution is  $768 \times 576$ , such as the one of Fig. 20. It was generated by the software *Poser*[26] and represents an articulated moving hand viewed from two close points of view (this sequence was given by the *MOVI* group from Inria Grenoble [1]). To obtain the clouds of points, we applied the same stereovision algorithm on this sequence than the one we are using on real sequences. The advantage of this sequence is to have no background, no blur, no noise nor distortion and a perfect calibration of the cameras. It allows us to test our algorithm on perfect data. As shown in Fig. 23, the tracking using the fully articulated model works quite good as long as there is no big occlusion.

The third one is the real sequence of Fig. 21, whose resolution is  $780 \times 582$  with a good framerate of around 30 frames per second. Despite partial occlusions during the sequence, the tracking of Fig. 22 is correct. It is not as perfect as the synthetic sequence because of not 100% perfect calibration of the cameras and because of little noise on pictures but the result is more than satisfying. We can conclude that our method works on bad quality data



image camera 1. image camera 2.

Figure 21: images extracted from the real sequence.

with a simplified model and on good quality data with the complex model.

The computing time used by the algorithm for a full model and dense clouds of points (a mean of 10.000 points) and around 100 Levenberg-Marquardt iterations per frame is 10s per image on an Intel processor 2.4GHz. This is not a very long time and it could be reduced by using less points (at least 50%) and less iterations (around 20 instead of 100).

We began to work on the number of points. The choice of which points to keep is yet arbitrary and we think of keeping a fixed number of points per voxel in order to have a more uniform repartition of the points. Reducing the difficulty of the problem will give an higher framerate but it does not seems to be able to be realtime in a close future. Expected performance are around half a second per frame.

## 6 Conclusion

We present in this report an hand tracking method based on an articulated model and 3-dimensional clouds of points computed from several cameras. In order to simplify this problem we filter the clouds of points (with colors and distance to the camera and to previous position of the hand) and constraint the model parameters (minimal and maximal value, linear dependency between several parameters). The improvement of the quality of the result thanks to the add of constraints is clearly visible. And the use of a good approximation of the distance instead of the simple one we used in the past gives noticable better results.

However, the results are not perfect, particularly when the cloud of points is not dense enough in the region of fingers. To improve results we consider several possibilities.

First, the search for a certain continuity from a frame to the next one (by filtering or smoothing) would prevent any jump during not so good tracked frame. It should allow a better estimation of the position of part of the hand when they are partially occluded. We are conscient that the drawback of such

a method is that it can be lost in case of rough inversion of direction of one finger during its occultation.

We are thinking to include the forearm in the model and in the tracking procedure in order to fix the wrist position and to prevent some sliding and some rotation of the model.

We could add some articular constraints in our model of the hand.

And finally we could use more the informations contained in the 2D images from the cameras. Currently they are only used to build the clouds of points and to filter the points using their skin color. It could be done by adding in the minimization algorithm the information on the distance between the contours of the model and the contours on the image as we did in [9].

The problem of the initial position is still open. We consider to ask the user to have a fixed position at the beginning of the sequence. We suggest an easy position such as an open hand, palm toward the camera.

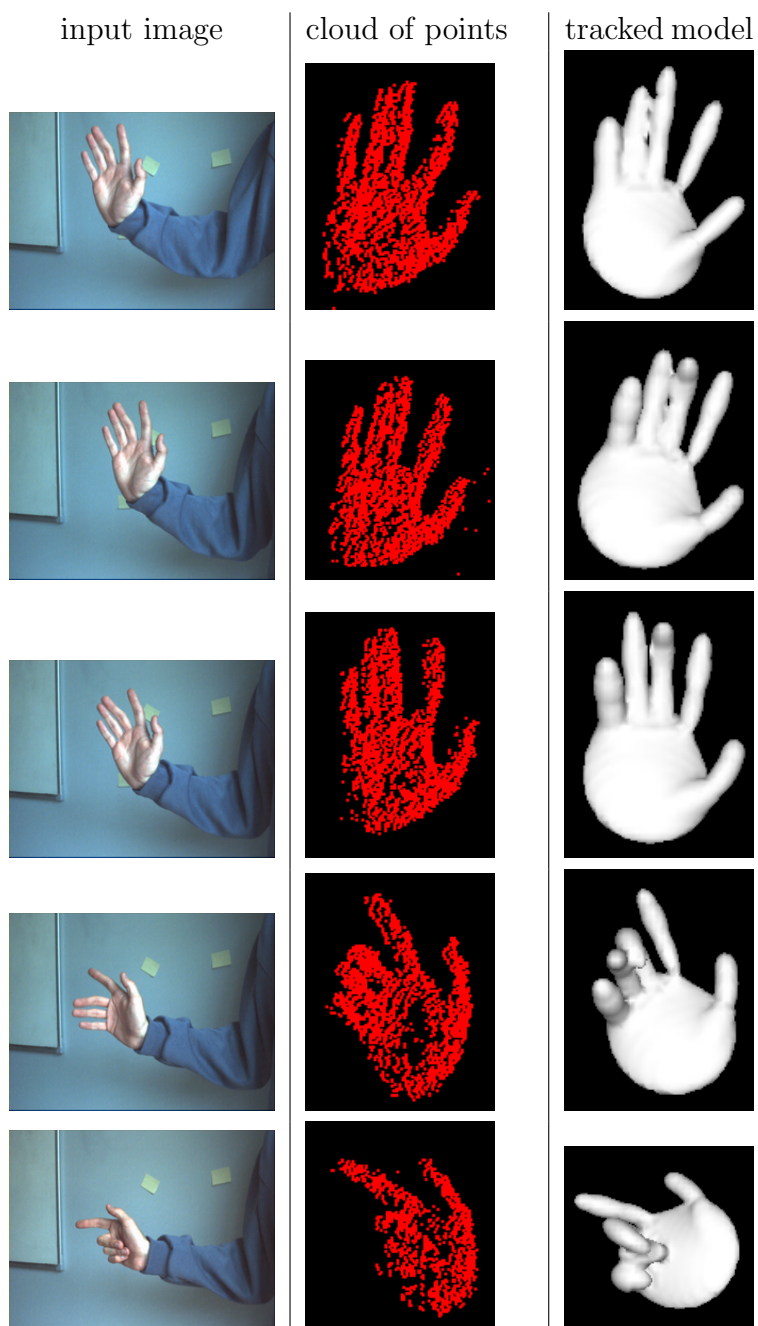
## References

- [1] G. Dewaele, F. Devernay and R. Horaud. Hand Motion from 3D Point Trajectories and a Smooth Surface Model. *European Conference on Computer Vision*, 2004.
- [2] Q. Delamarre and O. Faugeras. Finding Pose of Hand in video images: a stereo based approach. In *Proc. 3rd Int. Con. on Automatic Face and Gesture Recognition*, April 1998, pp. 585-590.
- [3] J. Rehg and T. Kanade. DigitEyes: Vision-Based Human Hand Tracking. *IEEE Workshop on Motion for Non-Rigid and Articulated Objects*, 1994, pp. 16-22.
- [4] R. Plankers and P. Fua. Tracking and Modeling People in Video Sequences. *Computer Vision and Image Understanding*, vol. 81, 2001, pp. 285-302.
- [5] Olivier Faugeras, Bernard Hotz, Hervé Mathieu, Thierry Viéville, Zhengyou Zhang, Pascal Fua, Eric Théron, Laurent Moll, Berry Gérard, Jean Vuillemin, Patrice Bertin and Catherine Proy. Real-time correlation based stereo: Algorithm, implementations and applications. *Rapport de Recherche 2013*, INRIA, 1993.
- [6] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. In *Proc. of the 14th Conference on Computer Graphics and Interactive Techniques*, 1987, pp. 163-169.
- [7] J. F. Blinn. A Generalization Of Algebraic Surface Drawing. *ACM Transactions on Graphics*, 1(3), 1982, pp. 235-256.
- [8] P.E. Danielsson. Euclidean distance mapping. *Computer Graphics and Image Processing*, volume 14, 1980, pp. 227-248.
- [9] F. Le Jeune. Suivi de la main dans une séquence vidéo. *Rapport de stage de DEA*, DEA IFA, Université de Marne-la-Vallée, France, 2001.
- [10] J.C. Terrillon, M. David and S. Akamatsu. Automatic detection of human faces in natural scene images by use of a skin color model and of invariant moments. In *Proc. of the Third International Conference on Automatic Face and Gesture Recognition*, Nara, Japan, 1998, pp. 112-117.
- [11] P. Fua. From Multiple Stereo Views to Multiple 3D Surfaces. *International Journal of Computer Vision*, vol. 24, August 1997, pp. 19-35.

- [12] P. Fua. A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features. *Machine Vision and Applications*, vol.6, Winter 1993, pp. 35–49.
- [13] T. Moeslund and E. Granum. A Survey of Computer Vision-Based Human Motion Capture. *Journal of Computer Vision and Image Understanding (CVIU)*, vol. 81, 2001, pp. 231-268.
- [14] W. T. Freeman and C. Weissman. Television control by hand gestures. *M. Bichsel, editor, Intl. Workshop on automatic face- and gesture-recognition*, Zurich, Switzerland, 1995. Dept. of Computer Science, University of Zurich, pp. 179-183.
- [15] T. Starner. Visual Recognition of American Sign Language Using Hidden Markov Models. *Master's Thesis*, MIT, Feb 1995, Program in Media Arts and Sciences, MIT Media Laboratory.
- [16] S. Lu, D. Metaxas, D. Samaras and J. Oliensis. Using Multiple Cues for Hand Tracking and Model Refinement. *Proceedings of Computer Vision and Pattern Recognition*, 2003.
- [17] N. Shimada, K. Kimura and Y. Shirai. Real-time 3D hand posture estimation based on 2D appearance retrieval using monocular camera. *Proc. 2nd Int'l Workshop Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems (RATFFG-RTS)*, Vancouver, Canada, July 2001, pp. 23-30.
- [18] A. Utsumi and J. Ohya. Multiple-hand-gesture tracking using multiple cameras. *Proc. CVPR*. Fort Collins, Colorado, June 1999, pp. 1473-1478.
- [19] K. Nirei, H. Saito, M. Mochimaru and S. Ozawa. Human hand tracking from binocular image sequences. *22th Int'l Conf. on Industrial Electronics, Control and Instrumentation*. Raipei, Aug. 1996, pp. 297-302.
- [20] Y. Wu, J. Y. Lin and T. S. Huang. Capturing natural hand articulation. *Proc. 8th International Conferance on Computer Vision*, Vancouver, Canada, 2001, pp.426-432.
- [21] B. Stenger, P. R. S. Mendonça and R. Cipolla. Model based 3D tracking of an articulated hand. *CVPR*, 2001, pp. 310-315.
- [22] W.H Press, S.A. Teukolsky, W.T. Vettering and B.P. Flannery. *Numerical Recipes in C++: The Art of Scientific Computing, Second Edition*. Cambridge University Press, New York, 2002.



- [23] Open Source Computer Vision Library. Intel.  
<http://www.intel.com/research/mrl/research/opencv/>
- [24] OpenGL. <http://www.opengl.org>
- [25] Firewire Camera. Point Grey Research Inc., Vancouver, Canada.  
<http://www.ptgrey.com/products/imaging.html>
- [26] Poser. <http://www.curiouslabs.com>



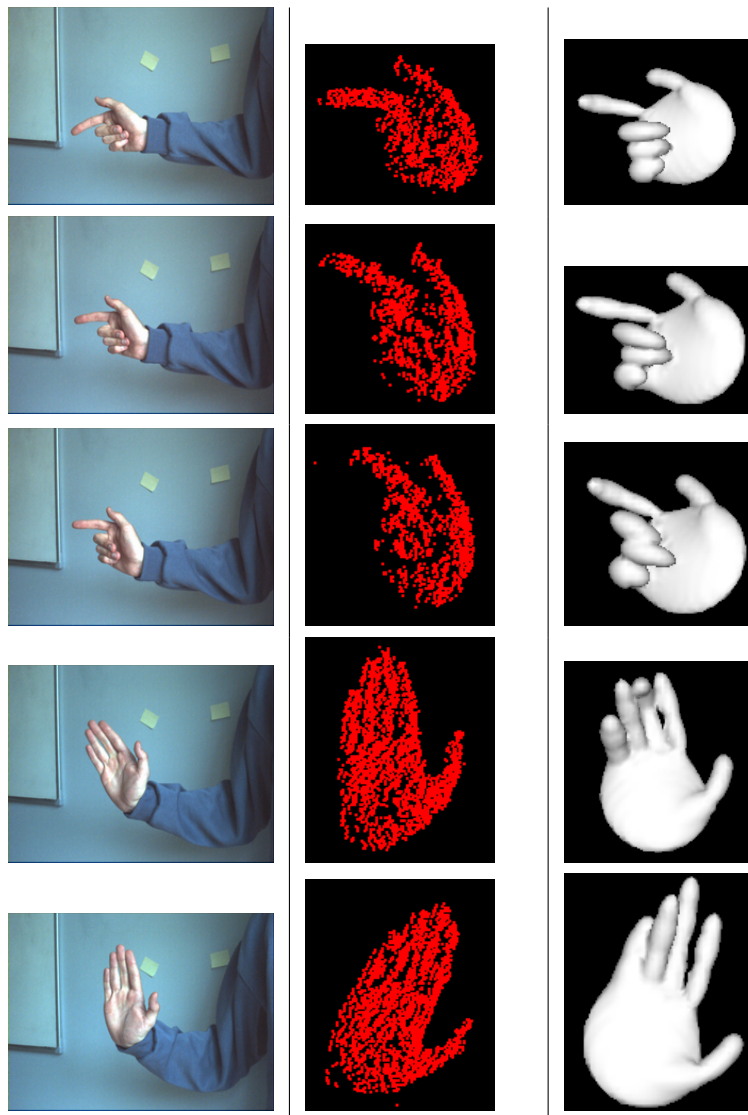


Figure 22: Tracking on a sequence with a fully articulated hand model (frame  $t = 0, 9, 18, 27, 36, 45, 54, 63, 72$  and  $81$ ).

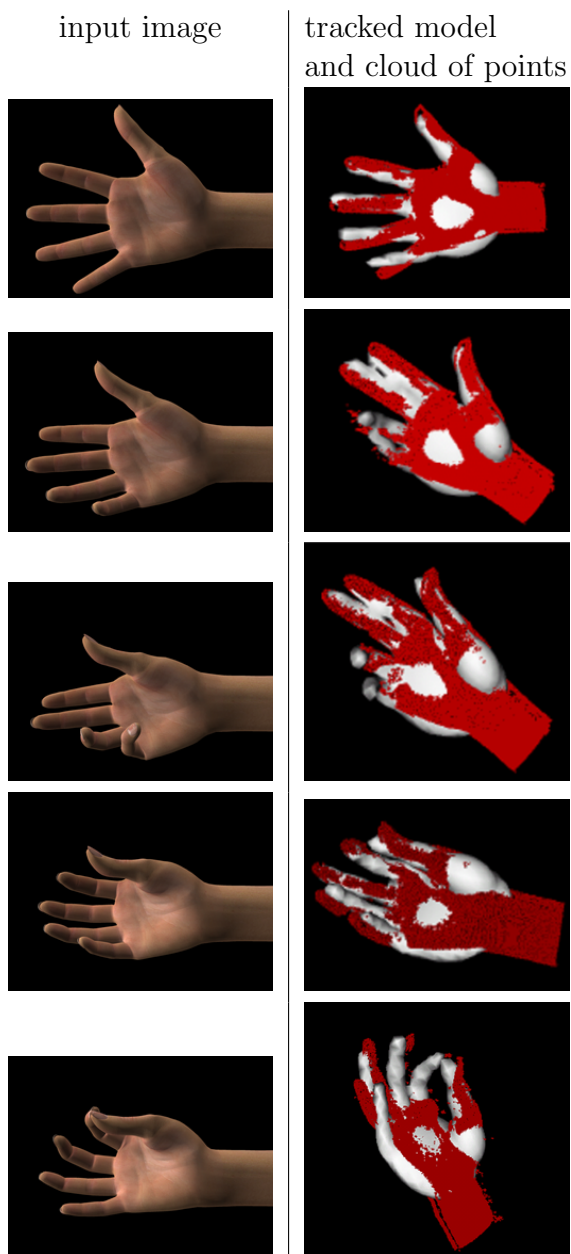


Figure 23: Tracking on a sequence with a fully articulated hand model (frame  $t = 0, 20, 40, 60$  and  $80$ ). On the left, input images, on the right, the cloud of point is red, the articulated model is grey.