Parallel Computing 34 (2008) 718–726

Contents lists available at ScienceDirect

Parallel Computing

journal homepage: www.elsevier.com/locate/parco

# A real-time full body tracking and humanoid animation system

C. Colombo *, A. Del Bimbo, A. Valli

*Dipartimento di Sistemi e Informatica, Università di Firenze, Via Santa Marta 3, I-50139 Firenze, Italy*

ARTICLE INFO

*Article history:*
Available online 19 September 2008

*Keywords:*
Human body tracking
Motion capture
H-Anim
Avatar

ABSTRACT

Non-intrusive human body tracking is a key issue in advanced human–computer interaction, with applications ranging from virtual reality to videoconference and telepresence. This paper describes a system for vision-based tracking of body posture. The system is explicitly designed to provide a robust yet simple and inexpensive solution to real-time body tracking through a careful choice of visual and kinematic models. Human posture representation is fully compatible with the MPEG-4 standard. Results of system application to a computer graphics scenario (animation of 3D avatars) are presented and discussed.



## 1. Introduction

The ability of computers to understand and interpret human actions is the basis for advanced man–machine interface design. Mice and keyboards will likely become obsolete as soon as new interaction devices will appear that can listen to and look at people as they express in everyday life. In particular, the possibility to capture the posture and movements of a person in real-time and in a non-intrusive way will open the scene to a new generation of applications in fields such as virtual and augmented reality, computer-supported cooperative work, computer-supported cooperative work, telepresence and robotics. Modern advanced interaction devices (e.g., 3D pointers or data helmets) do not appear to be immune to serious problems such as high intrusiveness, high cost, and low performance/cost ratio.

Being intrinsically non-intrusive and quite inexpensive, real-time computer vision is a technology that perfectly meets the basic requirements of next generation human–computer interaction applications. Concerning body motion capture, some vision-based approaches have been presented so far. In [1], color-based segmentation and contour analysis are used to track – even with a single camera – a human body represented by a set of connected blobs, homogeneous in color; template matching with four camera inputs is used instead in [2]; in [3], contour analysis is performed on images from three cameras disposed orthogonally; finally, disparity map analysis is carried out in [4], where special hardware is used to provide the map.

In this paper, an inexpensive and robust solution to body tracking in 3D using stereo vision is proposed, which is referred to as *Golem* system. An original system design choice is the deliberate search of simplicity through the use of affine camera models, standard image processing and sensible heuristics for the inverse kinematics model. Thanks to its low computational cost, the tracking process can be run in real-time as a secondary task, thus allowing heavy-weight primary tasks at the application level. As a test bed for such applications, a complete system for the real-time animation of 3D virtual characters is presented and discussed.

## 2. System overview

The system is designed to animate a virtual 3D puppet through body movements: when the user moves to a certain posture, the puppet replicates the same movement in its graphic workspace.

* Corresponding author.
*E-mail addresses:* colombo@dsi.unifi.it (C. Colombo), delbimbo@dsi.unifi.it (A. Del Bimbo), valli@dsi.unifi.it (A. Valli).

0167-8191/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.parco.2008.09.004

Fig. 1 shows the main system blocks, including in processing order (1) low level image processing and user body tracking on both left and right images; (2) stereo analysis and occlusion handling; (3) body pose reconstruction from stereo data via inverse kinematics and (4) virtual character update through computer graphics. The features tracked at the image level are the two feet, the two hands and the head; once these are identified and labeled independently in the two input images, a stereo procedure performing the triangulation of pairs of features with identical labels is carried out, also taking into account possible bad labelings due to occlusions. For pose reconstruction, a simplified human body representation and modeling derived from the MPEG-4 standard is used, by taking into account a limited number of basic degrees of freedom and then using a number of heuristic rules to infer the remaining ones. The computer graphics software is implemented on a Sgi workstation. Fig. 2 shows the various hardware pieces of the system and their functions and data exchange.

### 2.1. Communications

The three workstations communicate through TCP/IP stream sockets on a LAN or the Internet. In Fig. 3 the code execution is represented in time, top to bottom, and bright arrows represent data flow through sockets. The stereo computation algorithm requires that the two images of the user are grabbed instantaneously; to do this, the processing on the two vision workstations is synchronized using wait/go messages sent through the sockets just before the grabbing function. This software synchronization is sufficiently precise for the speed of common human movements. The rendering workstation instead

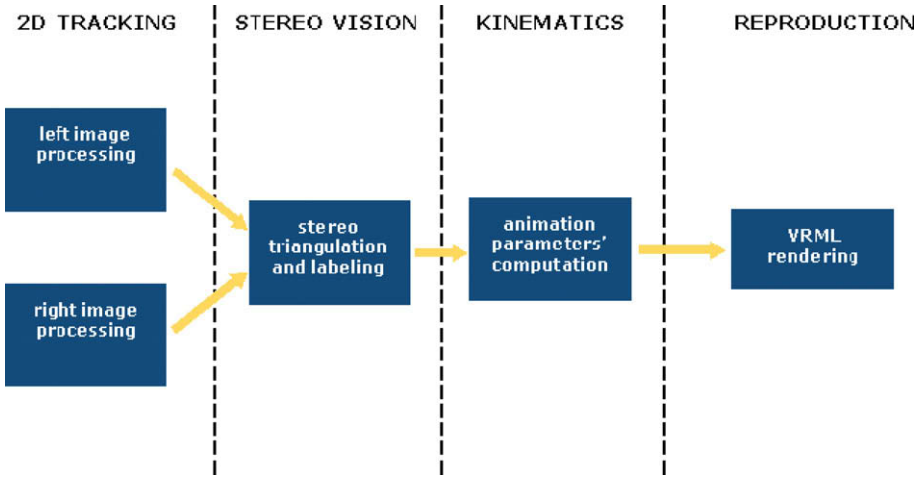


**Fig. 1.** The main system blocks.


**Fig. 2.** The various parts of the system.

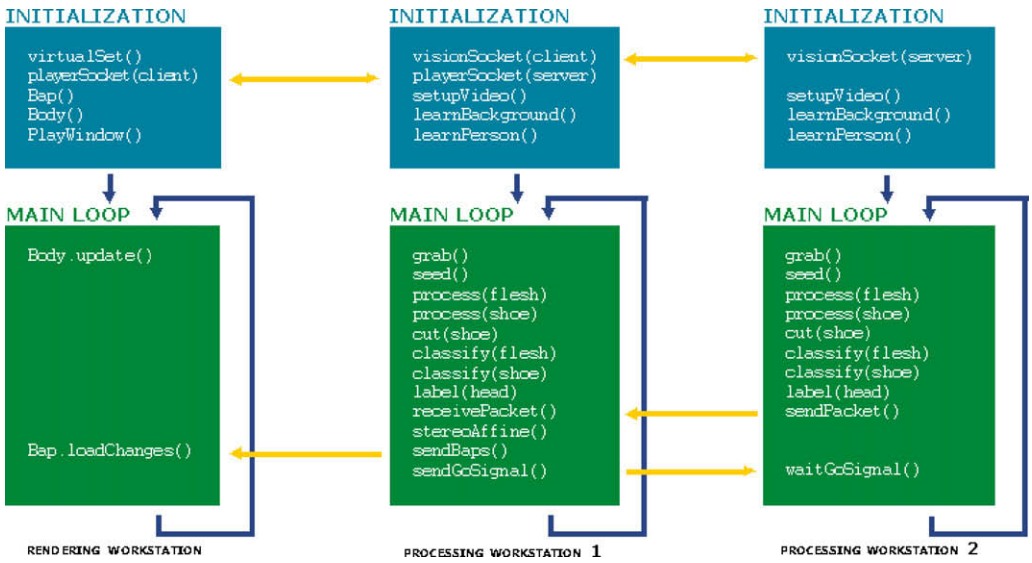


**Fig. 3.** Parallel execution and communications.

works asynchronously, updating the scene as new data is available. The main characteristics of the various system components are outlined in the following paragraphs.

## 3. The 2D tracker

Human body tracking at the image level is carried out using a fast multi-resolution algorithm mainly based on color information. The basic assumption is that the user wears shoes and pants of different colors, and that the only skin color parts visible are hands and face. The tracking algorithm is structured in two parts; the first one is a recursive routine that locates the foreground region by extracting in a single scan the whole body silhouette, the foreground/background separation (contour) pixels, and the regions belonging to the color classes of interest (shoes, face, etc.). In the second part the extracted pixels are further analyzed, and assigned to the various parts of the body to be tracked; also, possibly lost body parts are recovered in this phase. The algorithm can be summarized as follows:

1. *Single* recursive scan of the foreground
   - 1.1 extraction of the full silhouette
   - 1.2 extraction of contour pixels
   - 1.3 extraction of pixels with interesting colors
2. *Sequential* scan of extracted pixel properties
   - 2.1 contour analysis
   - 2.2 pixel labeling
   - 2.3 recovering of lost body parts

Fig. 4 illustrates the various phases of image-level body tracking (left image).

Video information ($160 \times 120$ pixel) is acquired directly in the *YUV* format, featuring a luminance signal $Y$ separated from chromatic information $(U, V)$. As the scene is assumed static, the only foreground element in the image is the person to be tracked. The background content is acquired once and for all at system startup. Then, until the person is found, the background image is analyzed at a very low resolution ($7 \times 5$ pixel) in search of foreground information; once this is found, a recursive region growing algorithm starts, operating at different resolution levels according to the color class each pixel belongs to. It is sufficient to grow the foreground region from a single point since the body is a connected shape. The growing algorithm expansion stops when the pixel encountered is very similar to the acquired background pixel in that position.

Interesting color classes are skin color and shoe color, which are read at startup, while the user stands in a fixed, reference position. Since the body silhouette is very smooth and regular, the algorithm only needs to expand in three directions (e.g., N, E, S-W) out of the 8-neighborhood canonical directions. To check whether the pixel belongs to foreground or background, or

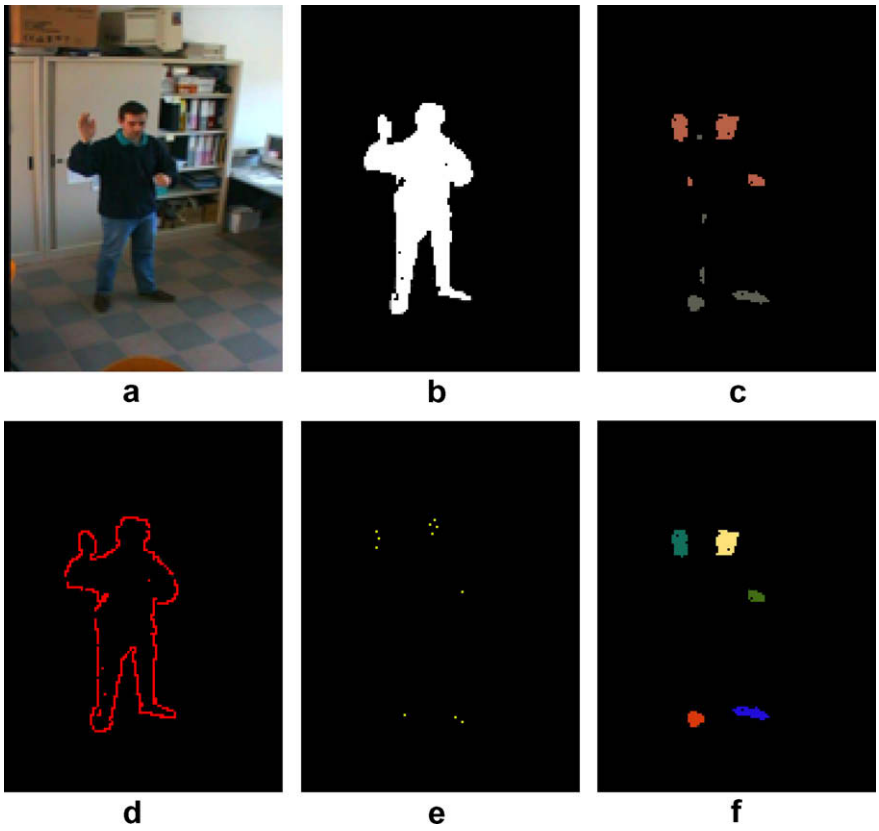


**Fig. 4.** (a): the original 160 × 120 image; (b): foreground region; (c): pixels belonging to color classes of interest; (d): silhouette contour; (e): hot spots from the contour; (f): labeled body parts.

whether it belongs or not to a certain color class, two different conditions are checked in two color spaces, $(Y, U, V)$ and $(U/Y, V/Y)$, the latter being very useful in dark image portions.

The pixel belongs to the foreground if the distance of his color from the color of the corresponding learned background pixel is over a certain threshold, in the normal color space *or* in the normalized one:

$$m_{\mathrm{FG}}(p,b) < \vartheta_{\mathrm{FG}} \vee m^{\star}_{\mathrm{FG}}(p,b) < \vartheta^{\star}_{\mathrm{FG}}, \tag{1}$$

else, it is part of the contour of the body, and is stored in a vector accordingly. To compute the distance in the color spaces used between these two colors, we use the following metric:

$$m_{\mathrm{FG}}(p,b) = \max(|Y_p - Y_b|, 4|U_p - U_b|, 4|V_p - V_b|), \tag{2}$$

where $p$ stands for pixel and $b$ stands for background. The chrominance distances are multiplied by four since the luminance variance is 3–4 times the chrominance variance, and a value of 4 has been used for computational efficiency, because dividing or multiplying integers by multiples of two can be done using bit shifts. The metric used to check if a pixel belongs to a color class is

$$m_{\mathrm{CC}}(p,c) = (Y_p - Y_c)^2 + 16[(U_p - U_c)^2 + (V_p - V_c)^2], \tag{3}$$

where $p$ stands for pixel and $c$ stands for color class. The pixel belongs to the color class if both the metric in the normal color space *and* in the normalized one are below two appropriate thresholds:

$$m_{\mathrm{CC}}(p,c) < \vartheta_{\mathrm{CC}} \wedge m^{\star}_{\mathrm{CC}}(p,c) < \vartheta^{\star}_{\mathrm{CC}}. \tag{4}$$

The multi-resolution nature of the algorithm allows a significant processing speed-up, as more than 90 percent of the raw image data are not analyzed at all, as shown in Fig. 5.

After region growing, topological filters are used in order to remove noise, region centroids are calculated, and the contour is scanned to spot short range curves, corresponding either to a limb or to the head.

Such hot points are used to properly initialize the region labeling process, which is based on association of foreground pixels to a sufficiently close tracked region.

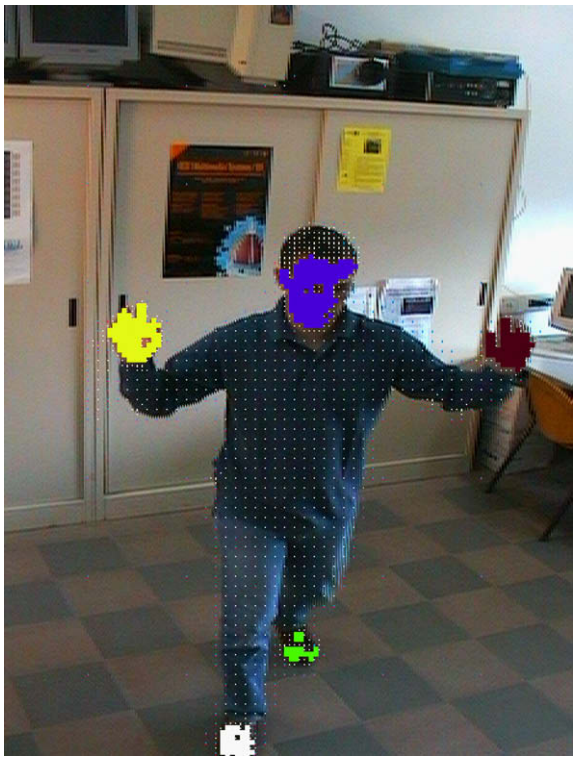

**Fig. 5.** Early vision processing.

If some body parts have been lost e.g. due to occlusions or fast movements, a new search begins; the algorithm looks for large clusters of remaining pixels, first looking near the locations eventually provided by the contour analyzer, providing recovery after self-occlusions or fast movements.

The main problem of this tracking approach are the possible labeling errors that can occur after a superimposition of parts having the same color.

The error recovery strategy adopted includes the use of heuristic considerations about the relative positions and areas of the body parts, comparison of region statistics (area, position, velocity) before and after occlusion and the exploitation of information encoded in the contour. A further decision step occurring at the level of stereo triangulation, when the information redundancy provided by the two cameras can be suitably exploited.

### *3.1. Occlusion handling: matching and labeling*

Correctly labeling the body parts belonging to the same color class introduces a new problem: indeed, in this case, labeling cannot rely on chromatic information. This phase is very important for matching the same body part in the two images and for labeling correctly the reconstructed 3D body parts. Previous history of limb movements is a relevant information for tracking: both in 2D and 3D the filtering and prediction of trajectories in the last few frames gives the first cue for tracking. This is sufficient when the body parts are quite far from each other in both image planes.

In case of occlusion this criterion is not sufficient; a set of statistics is matched before and after the occlusion, containing position (current and predicted), speed and area of the body part: the most probable match is chosen once the occlusion is finished. This method gives good results in many cases, but fails for complex motions, thus requiring an additional static check: since a wrong match gives non-realistic 3D reconstructed coordinates, the system checks if there is a big error for some frames, and in this case the labeling is changed.

## 4. 3D reconstruction

### *4.1. Symbolic stereo*

In order to evaluate the 3D point $(X, Y, Z)$ imaged at corresponding pixel locations $(u_l, v_l)$ and $(u_r, v_r)$ by the left and right cameras respectively, a stereo triangulation is carried out. This permits to compute the 3D locations of the main body parts (head, hands, feet). Since corresponding pixel pairs are not found by image search but are actually simply established by label matching, the triangulation algorithm can be referred to as *symbolic stereo*.

To significantly simplify computations, an affine camera model is adopted, according to which the scene-cameras transformation is linear [5]:

$$\begin{bmatrix} u_l \\ v_l \\ u_r \\ v_r \end{bmatrix} = \begin{bmatrix} p^l_{11} & p^l_{12} & p^l_{13} & p^l_{14} \\ p^l_{21} & p^l_{22} & p^l_{23} & p^l_{24} \\ p^r_{11} & p^r_{12} & p^r_{13} & p^r_{14} \\ p^r_{21} & p^r_{22} & p^r_{23} & p^r_{24} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

The above system is directly invertible and used to compute 3D points from image pairs once the camera parameters $p^l_{ij}$ and $p^r_{ij}$ are calibrated offline. To achieve this, a minimum of four known 3D points with corresponding image pairs are needed. Calibration is performed using standard routines [11]. The affine model is plausible if the region of interest in the scene has a small depth variation with respect to the distance from the camera. Model violations add a systematic 3D computational error, which is absolutely acceptable in the case of full body tracking with standard cameras.

### 4.2. Body model and kinematics

Once the 3D locations of the main body parts have been computed, a certain number of degrees of freedom must still be derived in order to animate a realistic humanoid model. The number of degrees of freedom to be solved depends on the complexity of the representation and the desired naturality of movement reproduction. In order to work in real-time, a simplified skeleton model was derived from the original MPEG-4 and H-Anim standards [6,7], modeling the human body as a hierarchic 3D skeleton composed of rigid links and rotational joints (see Fig. 6). In such models, the set of values for the angles and the

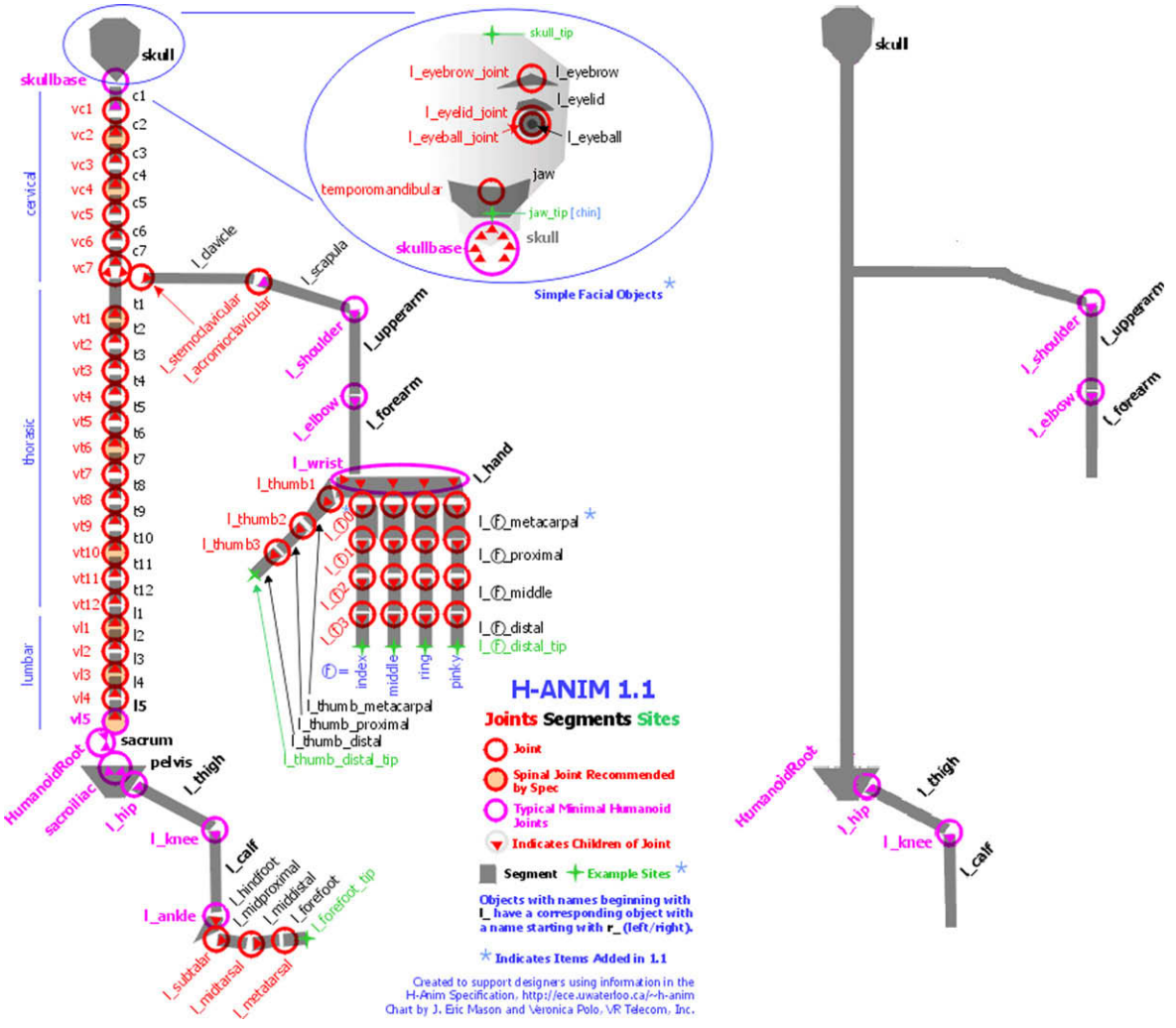


**Fig. 6.** *Left:* The model of the Humanoid Animation Working Group (H-Anim). *Right:* The reduced model used in this work.

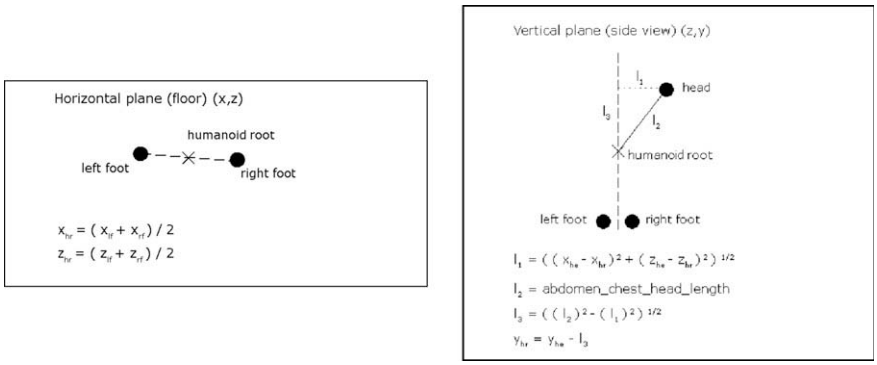


**Fig. 7.** Some of the heuristics used to solve the inverse kinematics problem.

translations of the joints completely defines the actor (and computer graphics character) posture. While the full MPEG-4 skeleton is made up of 89 joints and 186 degrees of freedom, the reduced model used in the current system implementation is restricted to 8 joints, 17 degrees of freedom and 9 rigid segments: head–chest–abdomen, upper and lower arms (left and right), upper and lower legs (left and right). Specifically, the degrees of freedom are: a 3D translation and rotation of the whole body, two rotations for hips and shoulders, and one rotation for knees and elbows (see Fig. 6).

Obtaining the internal and external parameters of the body model from the absolute 3D coordinates of the various body parts is an inverse kinematics problem. Such problem is usually addressed through a single mathematical framework regarding the desired body posture as a minimum-energy, iterative solution in an *N*-dimensional body configuration space [8]. A more real-time oriented alternative proposed in this paper is to decompose the full problem into smaller subproblems, which can be solved sequentially by means of suitable heuristics derived from well-known anthropometric rules. The first subproblem is solving for the external degrees of freedom (three translations and two rotations of the whole body) that fix the position of the head–chest–abdomen segment; subsequent problems are solving for the kinematic chains that depart from this segment: the two legs and the two arms. An example of heuristics is given in Fig. 7, illustrating the rule by which the so called *humanoid root* (i.e., the centroid of the whole body, located approximately at 45% from the ground on the vertical line through the feet centroid) can be evaluated and used as origin of a new body-centered coordinate frame to address the first subproblem. Once the head–chest–abdomen segment is located in 3D space (and consequently the shoulders and hips are located), the subproblems of arms and legs are solved using a simple analytic method based on trigonometry.

## 5. Experimental results

### 5.1. Setup

The user can move freely in an area of about $3 \times 3$ m; his 3D movements are tracked through two Sony EVI-D30 color cameras at a resolution of $160 \times 120$ pixels. The cameras are attached to the wall, at a distance of about 4 m from each other and from the user. Each camera signal is acquired and processed by an Sgi O2 workstation. The O2 devoted to the left image channel also performs stereo triangulation and inverse kinematics calculations. The computer graphics software is mainly based on the *Open Inventor* library package [10], and runs on an Sgi Octane workstation. Data communications among the three workstations are performed via socket connections.

During the tests, 21 users (15 males and 6 females) of different body sizes were told to move freely inside the environment, and were provided with visual feedback of their action. Specific tests were run in order to stress the system performance in presence of self-occluding gestures.

System failures are mostly related the occurrence of self-occlusions, causing breaking points in the tracking system to occur due to erroneous labeling.

Recovery after a breaking point is only possible if the user reaches an unambiguous posture, thus letting the second system to operate.

### 5.2. Processing time

The average frame rate of the basic system as measured during normal use tests is reported in Table 1. The lower frame rate in stereo mode is due to the delay for socket ethernet communications and synchronization. Despite the extra computational burden, the left channel workstation can work in parallel with the right one thanks to its doubled clock rate (300 MHz against 150 MHz). Due to the computational complexity required by graphic rendering, the Octane workstation runs at only half the speed of the others (12 frames per second).

**Table 1**
Frames per second on the two workstations

| | Mono | Stereo |
|---|---|---|
| Left channel | 28.9 | 24.3 |
| Right channel | 27.9 | 24.3 |

### 5.3. 3D reconstruction error

Another important factor to characterize the system is the precision of the 3D reconstruction. As anticipated earlier, the affine camera model introduces a systematic error, which is *independent* of the error on 2D measurements. To appreciate the influence of these two different error sources, the reconstruction error during body part tracking is reported in Table 2 both along a known trajectory at constant $Z$ (in which case the systematic error is null, the stereo computation solution being the same as with the full perspective model), and along a known trajectory at varying $Z$ (in which case a model error also arises). In the table, "rigid" movement refers to a body part translation without change in shape or orientation; "non-rigid" movement refers to a hand translating while opening and closing the fingers. In this last case the error is greater because the hand centroid moves in an unpredictable way.

The reconstruction error, even if not negligible, is nevertheless mainly of systematic nature, and quite tolerable in a number of interesting applications (such as the one presented in this paper) requiring a semi-qualitative rendering of human posture. The overall system performance can however be improved, if required, by introducing more sophisticated modeling both at the camera and at the kinematic levels.

### 5.4. Occlusion management

Table 3 reports on algorithm efficiency at tracking and matching in the presence of occlusions. As stated above, there are two anti-occlusion mechanisms that work together: the first one is immediate, and finds the most probable match just after the occlusion; the second one is a heuristic choice based on absolute observation, that acts continuously recovering the errors of the first system, normally when the posture becomes unoccluded again. Results are reported in terms of occlusion

**Table 2**
3D error at constant and varying depth (in cm)

| | *Constant Z* | | *Varying Z* | |
|---|---|---|---|---|
| | Rigid | Non-rigid | Rigid | Non-rigid |
| Head | 3.5 | – | 10.3 | – |
| Hand | 3.3 | 4.7 | 10.4 | 11.6 |
| Foot | 3.6 | – | 11.0 | – |

**Table 3**
Error recovery in case of occlusions (%)

| | First system | Second system |
|---|---|---|
| Head–hand | 98 | 2 |
| Hand–hand | 71 | 29 |

**Fig. 8.** The graphic character replicates user movements.

**Fig. 9.** Another character and posture.

recovery percentage with the two mechanisms. Notice that a complete (100%) recovery is achieved after the action of the second mechanism. Also notice that the most critical type of occlusion is the one reported in the second row (hand–hand): this is easily explained by the relative facility to distinguish the head from the hand based on the relative region areas.

### 5.5. Graphic rendering

Once the body animation parameters are calculated, these are sent through a socket connection to the rendering workstation, where a particular VRML [13] viewer developed for this work shows in real-time a H-Anim VRML computer graphics character moving in a VRML static scene, replicating the user's movements. The graphical interface of the viewer provides controls to change the point of view position and orientation in real-time using a mouse. Figs. 8 and 9 show the two views of the user taken by the system and the CG generated character in two different postures.

## 6. Conclusions and future work

The "minimalist" design of the system gives good results in recovering after self-occlusions, in precision of position measurement (with an error of a few centimeters), and speed.

Respect to the closest works, such as [9,2,12], in our system, the emphasis is more on full body 3D posture recovery and reproduction, and on the integration of several technologies into a single working prototype.

Future work will address a number of system improvements and extensions, e.g. allowing users to wear short clothes, the possibility to run without initial standing position for the person, color adaptivity to both foreground and background regions (to be computed partially through many frames), the possibility to track multiple people and objects, the introduction of variable step lengths for the multi-resolution algorithm based on the distance of the actor from the cameras and/or on his body area; this last strategy should help tracking the targets always at the same resolution, regardless of user position.

## References

[1] C. Wren, A. Azarbayejani, T. Darrell, A. Pentland, Pfinder: real-time tracking of the human body, IEEE Trans. PAMI 19 (7) (1997).
[2] T. Horprasert, I. Haritaoglu, C. Wren, D. Harwood, L. Davis, A. Pentland, real-time 3D motion capture, in: Proc. 1998 Workshop on Perceptual User Interfaces PUI'98, San Francisco, CA, 1998.
[3] S. Iwasawa, J. Ohya, K. Takahashi, T. Sakaguchi, S. Kawato, K. Ebihara, S. Morishima, Real-time, 3D estimation of human body postures from trinocular images, in: Proc. Workshop on Modelling People (mPeople), Corfu, Greece, 1999.
[4] N. Jojic, M. Turk, T. Huang, Tracking self-occluding articulated objects in dense disparity maps, in: Proc. International Conference on Computer Vision ICCV'99, Corfu, Greece, 1999.
[5] E. Trucco, A. Verri, Introductory Techniques for 3-D Computer Vision, Prentice-Hall, 1998.
[6] International Organisation for Standardisation, ISO/IEC JTC 1/SC 29/WG 11 N 2688, Information Technology – Generic Coding of Audio-Visual Objects – Part 2: Visual, Seoul, March 1999.
[7] B. Roehl (Humanoid Animation Working Group), H-Anim 1.1: Specification for a Standard Humanoid, August 1999.
[8] R. Boulic, R. Mas, Hierarchical kinematic behaviors for complex articulated figures, in: Thalmann, Magnenat-Thalmann (Eds.), Advanced Interactive Animation, Prentice-Hall Europe, 1996.
[9] A. Azarbayejani, A. Pentland, Real-time self-calibrating stereo person tracking using 3D shape estimation from blob features, in: ICPR'96 Vienna, Austria, August 1996.
[10] J. Wernecke, The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor, Release 2, Addison-Wesley, 1994.
[11] W.H. Press, S.A. Teukolsky, W.T. Vetterling, B.P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, Cambridge University Press, Cambridge, UK, 1992.
[12] P. Maes, B. Blumberg, T. Darrel, A. Pentland, The ALIVE system: full-body interaction with autonomous agents, Proceedings of Computer Animation 95, IEEE Press, 1995.
[13] R. Carey, G. Bell, C. Marrin, The virtual reality modeling language ISO/IEC 14772-1:1997, VRML Consortium (1997).