# Volumetric Modeling of 3D Human Pose from Multiple Video

Berend Berendsen[1]*, Xinghan Luo[2], Wolfgang Hürst[2], Remco C. Veltkamp[2]

[1] Department of Mediamatics, Delft University of Technology
Mekelweg 4, 2628CD Delft, The Netherlands
[2] Department of Information and Computing Sciences, Utrecht University
Padualaan 14, De Uithof, 3584CH Utrecht, The Netherlands
B.J.Berendsen@tudelft.nl, {xinghan, huerst, remco.veltkamp}@cs.uu.nl

**Abstract.** This paper describes a framework for modeling of 3D human pose from multiple calibrated cameras, which serves as the core part of a player pose-driven spatial game system. Firstly, by multi-view volumetric reconstruction, voxel-based human model is constructed. Secondly, by applying a hierarchical approach with a set of heuristics, fast indirect body model fitting algorithms are used to fit a predefined human model to the reconstructed data, and based on which human poses are modeled and semantically interpreted as certain control inputs to the game.

**Keywords:** Volume reconstruction, model fitting, tracking, pose semantics.

## 1 Introduction

We address the modeling of 3D human pose, by fitting a predefined articulated and parameterized body model to the volumetric human body that is reconstructed from multiple calibrated video cameras. Our prospective application is a player pose-driven spatial game, a new type of interactive computer entertainment. Without attaching any extra sensors, the players can control the game by attaining body poses in front of a set of multiple cameras connected to a gaming system. The core parts of such a system are the player pose modeling and semantics interpretation. For our prospective spatial game application, real-time algorithms rather than algorithms that find the perfect fit are preferred, to avoid unpleasant system response delay. Therefore we use an indirect body model fitting method with a hierarchical approach and common sense heuristics to increase speed.

**Related work.** Voxel-based 3D human model reconstruction [1, 2, 3] is recently recognized as a promising and robust method to recover human shape and motion features, which requires the multiple cameras to be calibrated. For fitting a predefined model to 3D reconstructed data, there are two common approaches: direct and indirect methods. The former is to fit sample points of the data to sample points of the template. To find a correspondence between two sets of points, several closest point and optimization algorithms can be used [2, 12]. The latter is to fit the template body

* Work done while at the Department of Information and Computing Sciences, Utrecht University.

parts around the volume or surface by finding feature points like the center of mass or the use of skin color voxels [3, 11]. Direct matching is more accurate than indirect matching, yet for a spatial game as application where system response speed depends significantly on the pose modeling efficiency, indirect matching is the better and faster solution.

**Contribution.** We have introduced a number of heuristics such as (i) using a mass seeking box in the torso to keep track of the torso and to be used as estimate for the spine, (ii) the use of the spine vector for finding the neck and check for consistent head direction, (iii) the use of line fitting for the initialization of the shoulders, and the use of clustering for tracking the shoulders, (iv) the placing of spheres on the shoulders and elbows for locating the upper arms and lower arms. In addition, we have systematically evaluated the efficacy of our method.

## 2 Volume Reconstruction

**Camera setup.** Our multiple video data are acquired by 4 AVT Marlin Firewire cameras (640×480, 25fps) mounted on 1.9-meter tripods. Camera synchronization is done by software. Because cameras opposite to each other provide the same (mirrored) silhouette, the 4 cameras are setup as shown in Fig. 1 (left), so that no camera is facing directly to any of the others.

**Calibration.** To specify the correlation between 3D lines in the world and 2D points in the four camera views, we use the respective Matlab camera calibration toolbox [4] and a 6×5 squares checkerboard as calibration object. The square corner coordinates are manually marked from the checkerboard images at different orientations. With the prior knowledge of the number and size of the squares on the checkerboard, the tool box calculates the intrinsic (Fig. 1 middle) and extrinsic parameters (Fig. 1 right) of a camera for each view, and align a global (world) coordinate system for the 4 cameras in this certain setup.

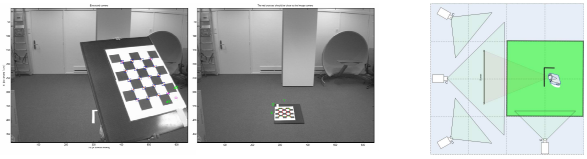

**Fig. 1.** Camera setup (*left*) and checkerboard calibration (*right*).

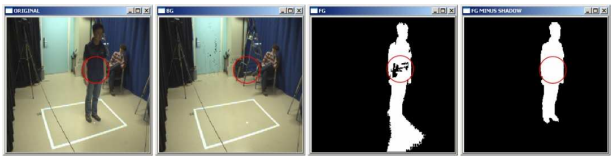

**Fig. 2.** (a) Video, (b) MoG background model, (c) foreground model, (d) shadow removal and region merging (red circle).

**Background subtraction.** For reconstructing the model of a person captured by multiple video cameras, first, the person in each video frame needs to be distinguished

from the background and extracted as a foreground mask [2]. We use the Mixture of Gaussians (MoG) method [5, 6] of OpenCV [9] for background subtraction and a simple shadow removal algorithm [7]. Pixel regions are merged according to a set of criteria: horizontal overlapping, X-distance of regions, summed area of merged regions [8]. See Fig. 2.

**3D voxel reconstruction.** The 2D silhouette points obtained by background subtraction are the projection of a person from 3D to 2D. This projection can be seen as a set of rays or a silhouette cone that contains the 3D points of the subject. The logical conjunction of the silhouette cones of all cameras results into a visual hull [2] that estimates of the subject's 3D shape (Fig. 3). We construct this visual hull using the voxel-based reconstruction method of Kehl et al. [2], resulting in a 3D voxel cloud model of a human subject. First a cube of voxels with a given depth (resolution) in millimeters is created. For each pixel of a camera view a lookup table (LUT) is initialized. To determine the correspondence between the pixels of each view and the voxels in the acquisition space, the center of each voxel is projected onto each view, and the voxel is added to the LUT corresponding to the pixel it was projected on. For each foreground image, an XOR operation detects changes in pixels. If a pixel has become part of the foreground, the corresponding voxels in the LUT are bitwise marked as visible for that view. Voxels that are part of the visual hull are visible from all views. They are selected by bitwise comparison of the LUT's entries.

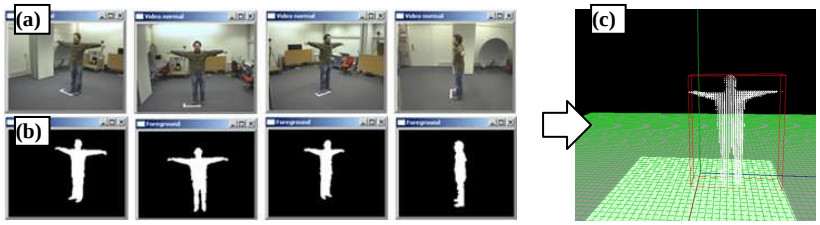


**Fig. 3.** (a) Four camera views, (b) corresponding silhouettes, (c) voxel reconstruction of visual hull.

## 3 Pose Modeling

**Human body model.** Our goal is to extract pose parameters and not model appearance. Therefore, we adopted a simplified generic articulated model for basic shape estimation of a human body and used a combined state vector [10] to parameterize the model. It consists of a 10-joint skeleton (stick figure) representing the basic human kinematic structure, based on which the torso, arms, and legs are modeled as cylinders and the head as a sphere (simple volumetric primitives). The human body parts of the model are defined in relation to body length. See in Fig. 4: Skeleton (left), combined skeleton and body parts (middle), and body parts (right).

**Body model fitting approach.** We use a fast indirect body model fitting methods similar to [3] and [11]. Template body parts are fitted around the voxel data based on feature points such as the center of mass, and voxels are then labeled to their

corresponding body parts. The fitting order follows a hierarchical approach with common sense heuristics: Due to its distinguished shape, first the head is located. Then the neck and pelvis points are found that determine the torso. From the torso the shoulders can be located, followed by the elbows and hands. In relation to the shoulders and pelvis, the hips are located, followed by knees and feet. This hierarchical approach requires multiple or reoccurring generic algorithms for each body model part. Therefore each fitting module has initialization, estimation, iterative refinement and validation steps as described in the following.

**Initialization and global tracking.** We use anthropometric measurements [3] to initialize the pose of the person in the scene while the subject is standing straight in a T-shape pose (Fig. 5). Using $L$ and $R$ to denote the length and radius of subsequent body part, and $L_{stat}$ for the length of the subject's stature, which is initialized by the height of the bounding box during the initialization pose. Hence all the other body parts can be initialized using $L_{stat}$ with the definition from [3]: $R_{head} \approx L_{stat}/16$, $L_{torso} \approx 3L_{stat}/8$, $L_{calf} \approx L_{stat}/4$, $L_{farm} \approx L_{stat}/6$, $L_{arm} \approx L_{stat}/6$, $L_{thight} \approx L_{stat}/4$. For the global tracking of the body model, a cube with dimensions $(2*R_{torso})^3$ is placed at the center of the torso $C_s$ that is determined during initialization. For each frame, the mass center of $C_s$ is optimized. The cube is large enough to remain within the torso somewhere along the spine, giving a global estimate of where the body is moving to.

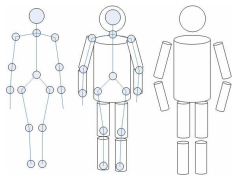

**Fig. 4.** Human body model

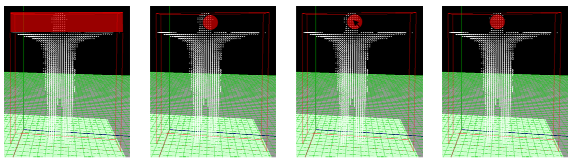

**Fig. 5.** Head initialization and refinement.

**Head fitting and tracking.** The head center is initialized by computing the centroid $C_h$ of the bounding box between $z = L_{stat}$ and $z = L_{stat} - 2*R_{head}$ and optimized in a subsequent refinement step (Fig. 5). To estimate $C_h$, the centroid $C_{ul}$ of the unlabeled head voxels within a sphere centered at the head centroid of the previous frame $C_{hp}$ is computed. From $C_{ul}$ and $C_h$, a displacement vector $V_d = C_{hp} - C_{ul}$ is calculated with magnitude: $d_m = (R_{head}/m_{hp}) \cdot m_{ul}$ ($m_{hp}$ stands for the previous number of marked head voxels, $m_{ul}$ for the current number of unlabeled voxels within the previous head sphere). The new position of the head is set as: $C_h = C_{hp} + \hat{V}_d \cdot d_m$. For validation, we look at the vector $V_s$ between the torso center $C_s$ and $C_h$: $V_s = C_h - C_s$. Since the head cannot move significantly into the torso, and the spine can not bend more than 90 degrees at the neck, the dot product between the spine vector and the head displacement vector between the previous and current head center cannot be negative, otherwise a re-estimate has to be made for $C_h$ by relocating the head along $V_s$ with a certain distance $d_{rl}$. The magnitude $d_s$ of the previous spine vector is $d_s = V_{sp} \cdot \hat{V}_s$. The displacement correction $d_c$ between $C_s$ and $C_{sp}$ is: $d_c = (V_s - V_{sp}) \cdot \hat{V}_s$. Using the relocation distance $d_{rl} = d_s - d_c$, head can be relocated along the spine: $C_h = C_s + \hat{V}_s \cdot d_{rl}$. In Fig. 6 (left to right): If $C_h$ moves into the torso, compute previous spine magnitude, compute displacement magnitude, and relocate head.

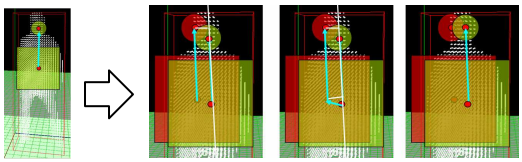

**Fig. 6.** Head validation and relocation.

**Torso fitting and tracking.** To initialize the torso cylinder, first the neck centroid $C_n$ in the cube centered at $C_h$ with z-value $L_{stat} - 2 * R_{head}$ is computed. Then the pelvis centroid $C_p$ in the bounding box between $L_{calf} + L_{thight}$ and $L_{calf} + L_{thight} + L_{torso} / 2$, the spine vector $V_s = C_n - C_s$, and pelvis position $C_p = C_n - \hat{V}_s \cdot L_{torso}$ are calculated (Fig. 7). The neck center is estimated by adding a vector with the reversed direction of the spine vector and the head radius $R_{head}$ as magnitude to the head centroid: $C_n = C_h - \hat{V}_s \cdot R_{head}$. The pelvis center is estimated by adding a vector in the same direction with the length of the torso to the neck position: $C_p = C_n - \hat{V}_s \cdot L_{torso}$. $C_n$ is refined by placing a sphere with radius $R_{head}$ on the estimated position. The center of mass of the non-head voxels is computed. $C_p$ is refined by placing the cap of the torso cylinder on the neck and the estimated pelvis position. By fixating the cap corresponding to the neck, and computing the center of mass $C_t$, the axis of the cylinder can be placed onto $C_t$. The process is repeated until $C_t$ has been stabilized. Finally, the voxels within the torso cylinder are marked as torso [3, 11].

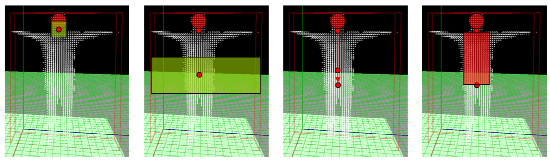

**Fig. 7.** Initialization of neck, pelvis, and torso.

To estimate the torso radius, firstly a 2D binary image is created by projecting the voxel slices along the spine vector between head and middle torso onto the z-plane while ignoring the head voxels. Least-squares line fitting is applied to the acquired 2D point set to determine the orientation of the torso. Secondly the voxels between the middle torso and pelvis are projected onto the z-plane. After the projection, the points are rotated in line with the orientation of the torso. The maximum x-width and y-height of the point blob now corresponds to the width and depth of the torso (Fig. 8), the average between these two values is then taken as torso diameter.


**Fig. 8.** Estimation of the torso radius.

**Arm/leg fitting and tracking.** To locate the arms, first the shoulder centroid $C_{sh}$ must be found. The unit vector $\hat{V}_o$ (collinear to the fitted line) is determined by means of least-squares fitting. The shoulder centroid can be computed as: $C_{sh} = C_n + \hat{V}_o \cdot R_{torso}$, or $C_{sh} = C_n - \hat{V}_o \cdot R_{torso}$ (mirrored to the other shoulder). The shoulder position is then found by placing a sphere with radius $R_{arm}$ on the initial estimate of the shoulder position. The centroid of the torso-marked voxels and the centroid of the unmarked voxels within the sphere are computed. The initial $C_{sh}$ is computed as the point in between both centroids. For estimation of the shoulder location, an outer cylinder is placed around the torso cylinder from the neck cap to the waist (Fig. 9 left). Then the unmarked voxels within the outer cylinder are selected. By applying k-means clustering on the selected voxels, two clusters of arm voxels are found (Fig. 9 right). For refinement, the two neck-to-cluster-center vectors are projected onto the upper cylinder cap (neck). The identification of the shoulders is done based on the distance between the current and the previous shoulder locations. As validation, the distance between these locations should not be larger than a threshold. Otherwise the displacement of the neck is applied on the previous shoulder position to determine the current shoulder position. Finally, the distance between the two shoulders should not be within a certain range. If so the shoulders are set apart.

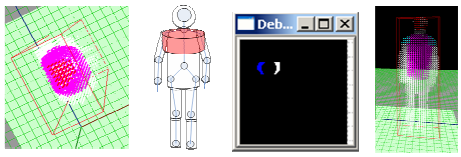


**Fig. 9.** Estimation of shoulder position.

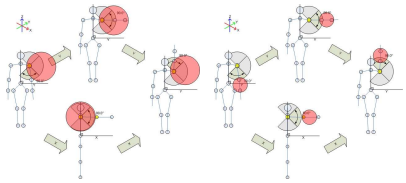

**Fig. 10.** Estimation of upper (*left*) and lower (*right*) arm direction.

To locate the elbows, an estimate of the direction of the upper arm is made by placing a sphere next to the shoulder $C_{sh}$ with radius $L_{arm}$ centered at $C_{sr} = \hat{V}_{sr} \cdot (R_{torso} - R_{arm})$, where $V_{sr} = C_{sh} - C_n$. Then, the centroid $C_{ed}$ of unmarked voxels is computed (Fig. 10 left). On $C_{ed}$, another sphere of radius $0.5 * L_{arm}$ is place to refine $C_{ed}$. By the direction from $C_{sh}$ to $C_{ed}$ and the magnitude of $L_{arm}$, the elbow position $C_e$ can be estimated as: $C_e = C_{sh} + \hat{V}_e \cdot L_{arm}$, where $V_e = C_{ed} - C_{sh}$. As last refinement step, a sphere with the radius of $R_{arm}$ is placed on $C_e$ to re-compute the centroid. If no voxels are found within radius $R_{arm}$, then the radius will be enlarged until a centroid is found. As validation, $C_e$ is compared to the previous elbow position. If their distance is larger than a threshold, the previous displacement of the elbow is applied. In the last step, the voxels within the upper arm cylinder are marked.

Similarly, the hand position is found by placing a sphere next to the elbow (in the opposite direction of the upper arm) with a radius of $0.5 * L_{farm}$ centered at $C_{srh} = C_e + \hat{V}_{srh} \cdot (L_{arm} - R_{farm})$, where $V_{srh} = C_e - C_{sh}$ (Fig 10 right). The centroid of the unmarked voxels within the sphere is then computed to find the middle of forearm. By the direction from $C_e$ to the found centroid and the magnitude of $L_{farm}$, the hand position is estimated as $C_{hd} = C_e + \hat{V}_{hd} \cdot L_{farm}$, where $V_{hd} = C_{hdd} - C_e$. Similarly to the elbow centroid refinement, a sphere with radius $R_{farm}$ is placed on $C_{hd}$ and the centroid of unmarked

voxels is computed. Finally the voxels within the forearm cylinder are marked as a hand.

To locate the legs, we use a similar approach as locating the arms.

**Pose semantics**. Depending on the specific game, the body pose can now be represented as a 5-tuple BodyPose = {left leg, right leg, left arm, right arm, torso}, where the legs and arms pose can have values {up, side, down}, and the torso pose can have values {up, forward}. For example, a Y-pose can be modeled as {down, down, up, up, up}, denoting a 'yes' input to the game, and a bowing pose can be modeled as {down, down, down, down, forward}, denoting exiting the game.

## 4 Evaluation and Conclusion

In order to evaluate the performance of our approach, we applied the body model fitting and pose modeling to three video sequences of different persons moving and performing several poses: two Asians (female and male) and one Caucasian (male). Assessment is done by subjective observation. From the initialization of the body model until the end of the sequence, one out of every twelve frames is evaluated with respect to the positions of head, torso, upper and lower arms. The position of the skeletal bone is verified in relation to the voxel and video data. If a bone fits onto the body part, it is marked as a good fit. If a bone is not exactly in but only up to half off, the supposed position is marked as a fair fit. In all other cases, it is marked as a poor fit. The evaluation of the first video sequence containing the Caucasian male is illustrated in Fig. 11 (left). For these frames, body model fitting works very well with almost completely correct matching for everything besides the lower arms. Errors there can be explained by: ‘sticky’ arms while the arms are very close to the torso and stick to it instead of fitting onto the voxel data corresponding to the lower arms; ‘float’ upper arm caused by voxels between the head and the arm; holes in voxel cloud; subject’s long hair etc. The overall fitting evaluation for all three video sequences is illustrated in Fig. 11 (right).

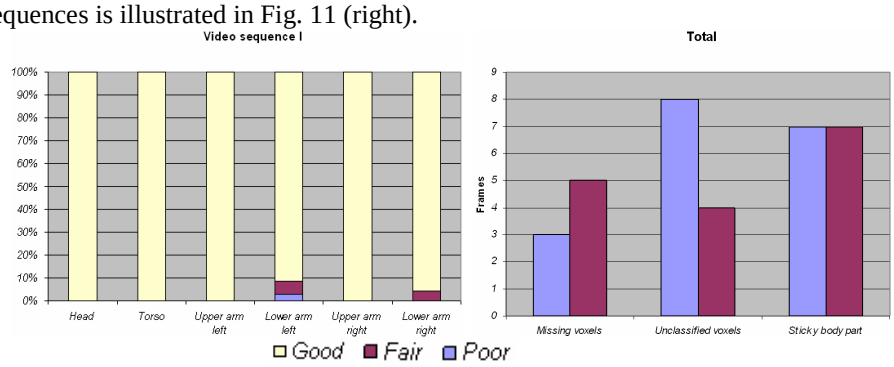


**Fig. 11.** Example video (*left*) and overall evaluation (*right*).

In this paper we described a framework for modeling of 3D human pose from multiple calibrated cameras (and pose semantics interpretation) as the core part of a spatial gaming system. By using more robust multi-view based 3D pose modeling,

our work opens the way to a more accurate semantics-level understanding of human poses.

For future work, body part constraints can be applied within the validation part of the algorithms to avoid impossible poses or movements of body parts. The evaluation can be extended by testing on more subjects. The framework will be extended to the multi-person case where mutual body occlusion and proximity must be handled.

**Acknowledgments.** This research has been supported by the GATE (Game Research for Training and Entertainment) project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

## References

1. Moeslund, T. B., Hilton, A., and Krüger, V. A survey of advances in vision-based human motion capture and analysis. International Journal of Computer Vision and Image Understanding. No. 104, pp. 90–126 (2006)
2. Kehl, R., Bray, M., and Gool, L. V. Markerless full body tracking by integrating multiple cues. In PHI'05 Workshop in Conjunction with ICCV05 (2005)
3. Michoud, B., Guillou, E., and Bouakaz, S. Real-time and markerless 3D human motion capture using multiple views. In 2nd Human Motion Workshop (2007)
4. Bouguet, J.-Y. Camera calibration toolbox for matlab (2007), http://www.vision.caltech.edu/bouguetj/calib_doc/
5. KaewTraKulPong, P., and Bowden, R. An improved adaptive background mixture model for real-time tracking with shadow detection. In 2nd European Workshop on Advanced Video-based Surveillance Systems (2001)
6. Stauffer, C., and Grimson, W. E. L. Adaptive background mixture models for real-time tracking. In CVPR 1999, pp. 2246–2252 (1999)
7. Porikli, F. Human body tracking by adaptive background models and meanshift analysis. In IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (2003)
8. McKenna, S. J., Jabri, S., Duric, Z., Rosenfeld, A., and Wechsler, H. Tracking groups of people. Computer Vision and Image Proceedings of AMDO02, Springer-Verlag, pp. 104–118 (2002)
9. Intel Open Source Computer Vision Library, http://www.intel.com/technology/computing/opencv/
10. Thalmann, D., and Magnenat-Thalmann, N. Handbook of Virtual Humans. John Wiley & Sons (2004)
11. Miki, I., Trivedi, M. M., Hunter, E., and Cosman, P. C. Human body model acquisition and motion capture using voxel data. In Proceedings of AMDO02, Springer-Verlag, pp. 104–118 (2002)
12. Tollmar, K., Demirdjian, D., and Darrell, T. Gesture + play exploring full-body navigation for virtual environments. In Proceedings of Computer Vision and Pattern Recognition for Human Computer Interaction (2003)