A Parallel Framework for Silhouette-based Human Motion Capture

Christian Theobalt, Joel Carranza, Marcus A. Magnor, Hans-Peter Seidel

Max-Planck-Institut für Informatik
Stuhlsatzenhausweg 85
66123 Saarbrücken, Germany
Email: {theobalt,carranza,magnor,hpseidel}@mpi-sb.mpg.de

Abstract

This paper presents a method to capture human motion from silhouettes of a person in multi-view video streams. Applying a hierarchical kinematic body model motion parameters are estimated by optimizing the overlap between the projected model silhouettes and the input silhouettes. The energy function driving the optimization is computed very efficiently using off-the-shelf graphics hardware. Exploiting the hierarchical structure of the human body, energy function evaluation is greatly sped up and a distributed implementation becomes feasible. Therefore, we present an algorithm for parallel silhouette-based motion capture employing multiple PCs and GPUs.

1 Introduction

Many conventional methods for estimating human motion parameters are intrusive, requiring optical markers or complex mechanical setups [16]. As a consequence, a separation of the generation of realistic motion from the generation of realistic physical appearance of the person is necessary. If the motion and appearance of the person need to be acquired simultaneously, such as in 3D video, no intrusion into the scene can be tolerated.

We have developed a method which non-intrusively estimates motion parameters using silhouette information [3, 21]. This method employs the use of a detailed geometric body model which, when combined with image-based rendering techniques, generates highly realistic images of a body in motion. The method optimizes the overlap between the projected model silhouette and all input silhouettes. The energy function that drives the optimization is efficiently computed using latest-generation graphics hardware.

Following the hierarchical structure of the human skeleton, the motion parameter estimation problem for the whole body can be decomposed into multiple smaller problems on kinematic sub-chains. Several of these sub-problems can be solved independently from each other. In this paper, we present several methods that exploit the compartmentalized nature of the problem to speed up motion parameter estimation. First, the energy function evaluation rate can be improved by only considering sub-windows in the image plane. Second, the rendering overhead can be significantly reduced by selectively rendering only those body parts that are currently optimized. Finally, the available computing environment can be optimally used by implementing the motion parameter estimation in a distributed system making use of several CPUs and GPUs.

The paper proceeds with a discussion of related work in Section 2, and a general overview of the proposed motion capture method in Section 3. The acquisition environment and the employed human body model are described in Sections 3.1 and 3.2, respectively. The energy function and its implementation in graphics hardware are described in Sections 4 and 4.1. Two techniques for speeding up the optimization, namely the application of a variable window size, and the pre-rendering of unchanged model-parts are described in Sections 4.2 and 4.3. The distributed computation is described in Section 5 and results with our system are given in Section 6. The paper concludes in Section 7 with an outlook to future work.

2 Related Work

In the Computer Vision literature, a variety of non-intrusive optical human motion capture techniques from video have been proposed (see [6] for a review). Some methods work on a single 2D image stream and apply, for example, frame differencing [11] or image skeletonization [8] to fit simple body models to human motion. Active contour models, i.e. deformable curves aligning with image discontinuities, have also been considered for 2D human body tracking [1]. An extension of a simple stick-figure model can be found in the *First Sight* system [13], where the 2D skeleton model is extended via ribbons to model volumetric body extent. In the *Pfinder* system [25], contingent image regions, called blobs, are represented by Gaussian distributions and tracked over time. In [2] optical flow information and the twist parameterization for rotations was used to capture human motion, the approach applies to multiple views too.

3D human motion capture approaches typically employ an explicit human body model consisting of a joint structure and some form of surface representation. Simple shape primitives, such as cylinders [9, 20] or superquadrics [7], are commonly used to represent limbs. The body models are fitted to the motion by aligning their projection with features in the image plane, such as image discontinuities. Inverse kinematics algorithms invert the model-to-image mapping to solve for the optimal parameters [27]. Physics-based approaches simulate forces to align the models with the image data[10], model deformations are also considered.

Silhouette information has demonstrated to be a powerful cue for the registration of texture information with 3D geometry [12]. The application of silhouette images for human motion capture has also been considered. In [5] a force field exerted by multiple image silhouettes aligns a 3D body model. In [18] a combination of stereo and silhouette information is used to fit a human body model. Recently, the application of reconstructed volumetric models (visual hulls) from silhouettes of a moving person for motion capture has also been considered. Ellipsoidal body models [4] , kinematic skeletons [14], or skeleton models with attached volume samples [23] are fitted to the volume data.

In 3D video, dynamic models of scenes that were recorded from several camera perspectives are reconstructed for re-rendering from novel viewpoints. The methods applied involve shape-from silhouette-like approaches, such as visual hull [15, 26] or stereo-based approaches [17]. In [3] the authors demonstrated that the combination of a non-intrusive human motion capture algorithm with a multi-view texture generation approach produces high quality free-viewpoint videos of human actors. Making use of graphics hardware, human motion parameters are estimated by maximizing the overlap between the projection of a template human body model and image silhouettes from multi-view video streams. The method presented in this paper extends this work by exploiting parallelism and by further improving the fitting times via decomposition of the whole optimization problem into smaller scale problems.

3 Overview

Motion parameters are estimated from a set of multi-view synchronized video streams recorded in our acquisition room [22]. Each of these video streams is transformed into a silhouette image by segmenting foreground and background. At every time instant, the set of input silhouettes defines an energy function which drives motion parameter estimation. We have specifically formulated our energy function so that it can be evaluated rapidly using commodity graphics hardware. We make use of a generic body model which is adapted to fit the actors dimensions and whose pose is determined by a set of 35 parameters which define rigid body transformations between body segments. The energy function measures how close a specific model pose is to the input images by comparing the difference between the input silhouette and a rendered model silhouette. The optimal pose is found using simple modifications of standard downhill energy minimization techniques (direction set method with Brent's line minimization [19]).

The optimization could easily be performed over the complete (and very large) parameter space. This approach, however, is not suitable since it often converges to incorrect local minima.

We have found that by breaking the optimization up into several sub-optimizations over lower dimensional parameter spaces, a-priori information about the problem can be exploited to increase both the speed and accuracy of motion parameter estimation. Techniques for accurate sub-optimizations were introduced in [3] which augmented the downhill optimization process with an efficient grid search. The grid search is a pre-processing technique that regularly samples a local neighbourhood in the parameter space to find a good starting point for the nonlinear minimization. A general overview of the optimization process for a given time instant is as follows:

Starting from the pose found in the last time instant.

1. Optimize torso position using standard downhill minimization.
2. Separately optimize each limb (arm/leg) pose using downhill minimization with grid search.
3. Separately optimize head, hands, and feet pose using standard downhill minimization.
4. Repeat procedure starting at (1) until convergence.

The use of this hierarchical optimization technique provides the opportunity to accurately estimate motion parameters extremely quickly. In this paper, we introduce two techniques to improve the energy function evaluation and demonstrate their application in a parallel implementation.

3.1 Acquisition

Our system uses synchronized multi-view video streams recorded in our camera studio as inputs. The studio [22] contains eight IEEE1394 cameras arranged at approximately equal angles and distances around the center of the room. The cameras are synchronized via an external trigger to record video frames at a resolution of 320x240 at 15fps. The use of an external trigger imposes a maximum frame rate of 15fps. Using Tsai's algorithm [24] the cameras' intrinsic and extrinsic parameters are determined, calibrating every camera into a common global coordinate system. The lighting conditions are controlled and all cameras are color-calibrated.

In each video frame, the foreground subject is segmented from the background via background subtraction. The algorithm employs per-pixel color-statistics to generate binary silhouettes (see [4] for details). Shadow regions that might lead to an incorrect classification of background pixels as foreground are eliminated via an additional angular threshold on pixel hue values.

3.2 Model

Our motion parameter estimation system makes use of a generic body model consisting of a hierarchic arrangement of 16 body segments (head, upper arm, torso etc.), each of which is represented by a closed triangle mesh. In total, the surface geometry consists of 21422 triangles. The model's kinematics are defined by a skeletal system consisting of 17 joints with interconnecting bone segments (Fig. 1). Rigid transformations at each of these joint locations define a specific body pose for the model. These transformations are constrained to imitate the actual motions of the body. Shoulder and hip joints are represented by 3 degree-of-freedom (DOF) ball joints and elbow and knee joints are represented by 1-DOF hinge joints. Assuming the actor stands in a specific initialization pose, the generic body model shape is conformed to that of the person through a silhouette-based fitting process [3]. From this point on, bone lengths and segment geometry is fixed and motion parameter estimation is performed by energy function minimization.

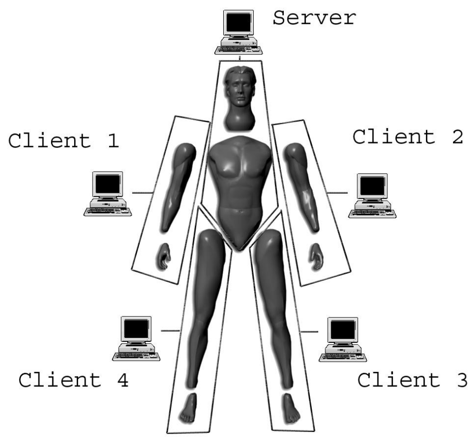

Figure 1: Exploded view of the body model and assignment of different kinematic sub-chains to computers in the distributed client server setup.

4 Energy Function

The energy function is a score of how closely a body model in a specific pose corresponds to the input images at a given time instant. In order to estimate motion parameters, for each time frame, the pose parameters are varied until they converge to some minimum of the energy function. We make use of standard techniques for downhill minimization. Naturally, this requires a significant number of function evaluations, and thus it is absolutely necessary to formulate an energy function which is both accurate and efficient.

[3] demonstrated that robust parameter estimation can be accomplished by computing an XOR between input silhouettes and rendered model silhouettes. Each camera has an input silhouette which is generated by background subtraction and is constant for a given time instant. To evaluate how closely a given model pose conforms to that input silhouette, the model is rendered as a silhouette and the two images are XOR'ed. The score for that camera is the sum of all non-zero pixels in the XOR result. The energy function result for a given pose is the sum of results from all eight camera views.

Energy function evaluation can be implemented efficiently by using commodity graphics hardware. Each input silhouette is packed into a bit plane of a byte sized buffer. That buffer is transferred into the OpenGL stencil buffer, and successive drawings of the model compute the XOR in each bit plane. The buffer is then loaded back into main memory and bit-counting is performed on the CPU. Having 8 cameras and an 8-bit stencil buffer, each function evaluation requires the rendering of the model from each camera perspective, but only requires a single read and write to the stencil buffer to compute the final value. The major bottleneck in our system is the evaluation of the energy function. Evaluation speed, and thus motion parameter estimation speed, can be increased significantly by reducing the amount of information that must be transferred to and from the GPU.

4.1 Implementation in Hardware

An understanding of the internal implementation of the XOR function is critical to understanding Section 4.3 and so is presented here. The stencil buffer stores 8-bit values which can by modified through a number of simple operations on a per-fragment basis. To compute an XOR in the stencil buffer, the model is rendered from each camera perspective into that camera's bit plane. Each fragment which passes the depth test is told to invert the bit at its corresponding pixel. In order to prevent multiple fragments from inverting a single pixel more than once, the camera's projection matrix is modified so that all vertices project into z=0 plane. With the depth test properly set to reject all fragments with a z-value larger or equal to the value in the depth buffer, at most one inversion occurs per pixel. Once the XOR has been computed for all 8 cameras, the stencil buffer is transferred back to the CPU which counts the total number of bits that are set.

4.2 Variable Window Size

The partitioning of the motion parameter estimation process into a number of sub-optimizations over specific body parts means that in any given optimization, only one body part is moving at a time. The energy function evaluation can be restricted to the region of the image plane in which the body part is moving. This body part may take up only a small portion of the overall silhouette. An arm or leg is much smaller than a person's torso, and a hand or foot is much smaller than an arm or leg. Therefore, there is no need to transfer the entire camera silhouette to the GPU. For each body part in the hierarchy, a fixed size window is chosen over which to evaluate the energy function. When that body part's pose is being estimated, only that window of the input silhouette is transferred to and from the GPU. The location of this window is taken to be the center of mass of the body part in the pose estimated from the last time instant (see Fig. 3). Consequently, the window location, and thus the silhouette information sent to the GPU, does not vary for a given time instant.

A reduced window size allows for rapid evaluation of the energy function for small body parts. As such, the choice of window sizes is critical to both performance and accuracy. The projected size of a body part can vary greatly depending on position. Obviously, in some extreme cases (for example the person goes and puts his face in front of a certain camera), a body part may exceed the window size in a specific camera. However, it is worth noting that with a reasonably camera setup and window sizes, even if a body part exceeds the window of a certain camera, it will certainly fall entirely within the window of several other cameras, and thus its pose still be reliably estimated. Regardless of this fact, we choose our window size very conservatively (128x128 pixels for arms and legs, 64x64 pixels for head, hands and feet).

Estimating the torso pose results in motion of all body parts. The model silhouette varies over a large region making it impossible to choose a significantly reduced-size window. However, we have found that the torso and its linked body parts are large enough to be reliably tracked with silhouette

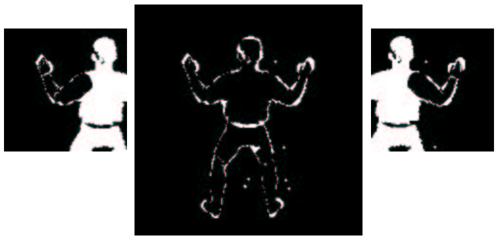

Figure 3: Global energy function (center) and smaller sub-windows (128x128 pixels) used to optimize the arm positions (shown for one camera).

images at half-resolution. We simply use the silhouette images scaled down to 160x120 resolution for estimating the pose of the torso.

4.3 Body Part Pre-rendering

The energy function evaluation rate can be further sped up by reducing the number of geometric primitives that need to be rendered each time. During optimization of a limb, for example, the pose parameters of all the other body parts are not modified, hence their projection into all the camera views does not change. The energy function evaluation speed can therefore be greatly improved by only rendering the geometry of those body parts that are currently optimized. The problem with this approach is that it adds a wrong contribution to the XOR energy function. During computation of the XOR in the stencil buffer the bits are set in those regions where there are pixels from the image silhouettes, but where no body part projects to, since it was excluded from rendering. To eliminate this erroneous contribution to the energy function, an additional *pre-rendering* step needs to be performed which creates a mask that corrects the error function on the CPU. For each camera view, this mask contains a 0-bit for each pixel to which a body part projects that does not change during optimization, and a 1-bit for all other pixels. The masks are generated by setting the stencil buffer configuration appropriately and rendering the model without the body parts that are optimized from each camera view. The energy function error is corrected on the CPU by computing a pixel-wise AND between the stencil buffer bit-planes and all camera masks before counting the set bits. Figure 2 illustrates the modified error function evaluation.

5 Distributed Computation

The compartmentalized nature of the problem suggests that a distributed computation approach is feasible. The optimization of a specific body part is primarily dependent on the optimization of body parts which are higher in the hierarchy and relatively independent of any other body parts. For example, the correct position for the left arm is unaffected by the position of the right arm and legs.

Whereas the redefinition of the energy function minimizes the amount of information travelling across the GPU bus, running a distributed computation model effectively increases the bandwidth of a single ”virtual” bus. The task of motion parameter estimation is split among five computers (each with a high-performance GPU) in our distributed system.

A single computer, designated as the server, is responsible for estimating the position of the torso and head, while each of the four clients’ task is to estimate the position of the limb and attached hand or foot. The computers are connected over standard 100 MBit/s network connections and communicate with a very basic protocol over TCP/IP.

Motion parameter estimation at each time step begins at the server who packs the input silhouettes into a single buffer and then transfers this information to the clients. The server optimizes the position and rotation of the torso and then sends the resulting model pose to the clients. At this point, each client begins estimating the motion parameters for its respective limb and extremity, while the server estimates the pose parameters of the head. In this way 29 out of the 35 parameters are estimated concurrently over 5 GPUs (Fig. 1). Once each client completes its pose estimation, its results are transferred back to the server, which, after receiving all results, will iterate over this time step again and refine the estimates, or move on to the next time step. The grid search used by each limb to robustly estimate pose parameters is a significant portion of the computation in the larger pose estimation process, and so by performing all limb fitting concurrently, a significant speedup is expected.

Certainly, other models for distributed computation exist. It is feasible to subdivide the optimization of a given body part up among several GPUs as well. However, we chose this model for our system because of its proper balance of speed, simplicity, and hardware requirements. Introduc-

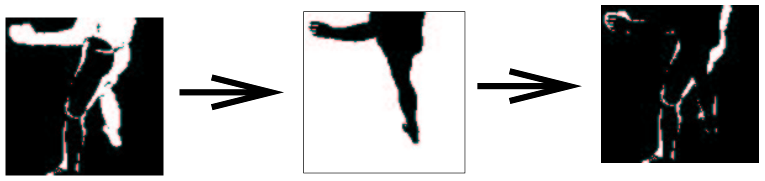

Figure 2: Body part pre-rendering: During the motion parameter estimation of a kinematic sub-chain, only the segments of this sub-chain are rendered. To correct for errors in the XOR energy function, a mask is pre-computed before the optimization starts. A bitwise AND between the mask and the stencil buffer is computed to get the final value of the energy function. The same process applies to all bit-planes of the stencil buffer.

ing the additional complexity of several computers per body part would provide relatively minor speed improvements in comparison to the speed improvements of using a single computer versus five. The investment in such significant additional hardware for small speed increases is rarely desirable. The use of five systems is well suited for our purposes, as four computers are needed anyway to control our camera setup (see Sect. 3.1).

Our motion capture results do not deteriorate using our distributed system. With some rare exceptions, the estimates obtained for each limb or extremity are completely independent of that of other limbs. This is because, for a large majority of poses, the limbs are distinct from each other in at least one camera view. The situation one would expect to be problematic for distributed motion parameter estimation, namely where there is no distinction between two limbs in any camera view, is a fundamental problem for any silhouette based method. Fortunately, such poses (for example a person in the fetal position) are quite uncommon.

6 Results

The impact of our newly introduced methods on function evaluation times are presented in table 1. All the computers used for our tests feature a 1.8 GHZ Intel™ Xeon CPU, 512 MB RAM and an Nvidia GeForce3™ GPU.

The XOR energy function gains a significant performance increase by decreasing the window size. Nevertheless, it is evident that as the window size decreases, rendering the model becomes a significant bottleneck. XORPR uses the silhouette subtraction energy function in combination with pre-rendering to reduce the rendering time and provides extremely good results. Observe that the XORPR method gains significant speed compared to XOR for smaller window sizes since during the optimization of the arms (128x128 window) or the head and the hands (64x64 window) most of the model geometry is excluded from rendering.

Window Size	XOR	XORPR
320x240 (full)	95.9	95.5
160x120 (half-res)	131.1	131.2
128x128 (arm)	133.7	433.1
64x64 (head)	144.9	855.4

XOR - original method
XORPR - XOR with pre-rendering

Table 1: Energy function evaluations per second for different stencil window sizes using one computer.

We evaluated our system on two different video sequences. Sequence A shows a person exhibiting a series of basic motions at slow speeds. Sequence B shows a ballet dancer performing a number of complex dance motions at high speed. To accurately estimate the motion parameters, the size of the grid search for arms and legs in sequence B had to be set much larger than that of sequence A. Both sequences were fairly long, roughly 400 frames in length. The motion parameters were estimated for both of these sequences in three different ways, XOR on a single computer, XORPR on a single computer and XORPR in a parallel system using

	Seq. A	Seq. B
XOR	7.98	14.1
Single Client	3.30	10.1
Distributed	1.16	1.76

XOR - original method with single computer
Single Client - XORPR with single computer
Distributed - XORPR with 5 computers

Table 2: Average fitting time per frame (s).

5 computers. The methods introduced in this paper increase the speed of optimization by almost an order of magnitude. Table 2 shows the average fitting times we obtained for both sequences with the different ways of parameter estimation. Whereas the methods presented in this paper already produce a significant speedup if only one computer is used for motion estimation, the parallel implementation leads to even faster fitting times. Fig. 4 shows two input video frames and the corresponding recovered body pose from sequence A, in Fig. 5 the same is shown for sequence B. The silhouette-based motion estimation approach can robustly capture a large range of human motion. Even complex twists and turns of the human body and very fast movements are reliably recovered. In comparison to the results presented in [3] the results with the new approach don't deteriorate, and the fitting times are by orders of magnitude smaller.

7 Conclusion and Future Work

We have presented a distributed system which makes use of multiple CPUs and GPUs to rapidly and accurately estimate motion parameters using a detailed body model. The major bottlenecks that limit the performance of the motion parameter estimation, namely the memory transfers to and from the graphics board, as well as the rendering overhead for the body geometry, were identified. It has been demonstrated that the influence of these bottlenecks on the performance can be reduced by limiting the energy function evaluation to sub-windows of the image plane and by pre-rendering unchanging body-parts. The fitting times can be further improved by solving for the motion parameters in a parallel system.

In our future work, we plan to incorporate texture information into the tracking process and will investigate the application of a single-skin body model.

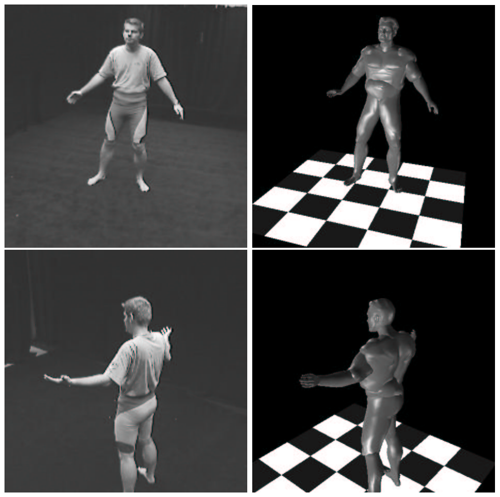

Figure 4: Input views and corresponding captured body poses from sequence A.

References

[1] A. Baumberg and D. Hogg. An efficient method for contour tracking using active shape models. In *Proc. of IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 194–199, 1994.

[2] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Proc. of CVPR 98*, pages 8–15, 1998.

[3] J. Carranza, C. Theobalt, M. Magnor, and H.P. Seidel. Free-viewpoint video of human actors. In *Proc. of SIGGRAPH2003*, to appear, pages 569–577, San Diego, USA, 2003. ACM.

[4] K.M. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. In *Proc. of CVPR*, volume 2, pages 714 – 720, June 2000.

[5] Q. Delamarre and O. Faugeras. 3D articulated models and multi-view tracking with silhouettes. In *Proc. of ICCV'99*, pages 716–721, 1999.

[6] D.M. Gavrila. The visual analysis of human movement. *CVIU*, 73(1):82–98, January 1999.

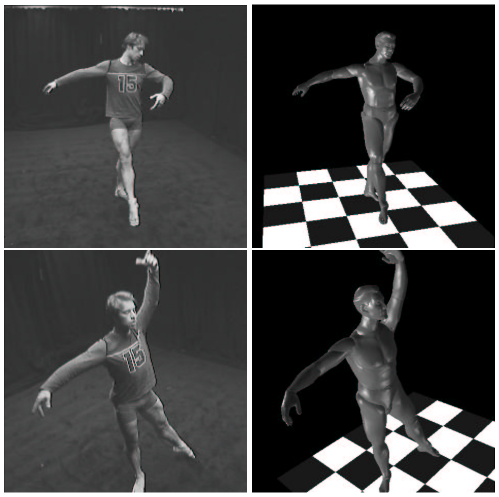

Figure 5: Input views and corresponding captured body poses from sequence B.

[7] D.M. Gavrila and L.S. Davis. 3D model-based tracking of humans in action: A multi-view approach. In *Proc. of CVPR*, pages 73–80, 1996.

[8] Y. Guo, G. Xu, and S. Tsuji. Tracking human body motion based on a stick-figure model. *Journal of Visual Communication and Image Representation*, 5(1):1–9, 1994.

[9] D. Hogg. Model-based vision : a program to see a walking person. *Image and Vision Computing*, 1(1):5–20, 1983.

[10] I. A. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection. In *Proc. CVPR*, pages 81–87, 1996.

[11] Y. Kameda, M. Minoh, and K. Ikeda. Three dimensional motion estimation of a human body using a difference image sequence. In *Proc. of ACCV'95*, pages 181–185, 1995.

[12] H. Lensch, W. Heidrich, and H. P. Seidel. A silhouette-based algorithm for texture registration and stitching. *Graphical Models*, 64(3):245–262, 2001.

[13] M. Leung and Y. Yang. First sight : A human body outline labeling system. *PAMI*, 17(4):359–379, 1995.

[14] J. Luck and D. Small. Real-time markerless motion tracking using linked kinematic chains. In *Proc. of CVPRIP02*, 2002.

[15] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proc. of EGRW'01*, pages 116–126, 2001.

[16] A. Menache. *Understanding Motion Capture for Computer Animation and Video Games*. Morgan Kaufmann, 1995.

[17] P.J. Narayanan, P. Rander, and T. Kanade. Constructing virtual worlds using dense stereo. In *Proc. of ICCV 98*, pages 3 – 10, 1998.

[18] R. Plaenkers and P. Fua. Tracking and modeling people in video sequences. *CVIU*, 81(3):285–302, March 2001.

[19] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes*. Cambridge University Press, 1992.

[20] K. Rohr. Incremental recognition of pedestrians from image sequences. In *Proc. of CVPR 93*, pages 8–13, 1993.

[21] C. Theobalt, J. Carranza, M. Magnor, and H.P. Seidel. Enhancing silhouette-based human motion capture with 3D motion fields. In *Proc. of Pacific Graphics*, to appear, page NN, 2003.

[22] C. Theobalt, M. Li, M. Magnor, and H.P. Seidel. A flexible and versatile studio for synchronized multi-view video recording 2003. In *Proc. of VVG*, pages 9–16, 2003.

[23] C. Theobalt, M. Magnor, P. Schueler, and H.P. Seidel. Combining 2D feature tracking and volume reconstruction for online video-based human motion capture. In *Proc. of Pacific Graphics 2002*, pages 96–103, 2002.

[24] R.Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. of CVPR'86*, pages 364–374, June 1986.

[25] C.R. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *PAMI*, 19(7):780–785, 1997.

[26] S. Wuermlin, E. Lamboray, O.G. Staadt, and M.H. Gross. 3D video recorder. In *Proc. of Pacific Graphics 2002*, pages 325–334, 2002.

[27] S. Yonemoto, D. Arita, and R. Taniguchi. Real-time human motion analysis and ik-based human figure control. In *Proc. of IEEE Workshop on Human Motion*, pages 149–154, 2000.