Automated Body Modeling from Video Sequences

Ralf Plänkers and Pascal Fua
Computer Graphics Lab (LIG)
Swiss Federal Institute of Technology
CH 1015 Lausanne (EPFL)
{Ralf.Plaenkers,Pascal.Fua}@epfl.ch
http://ligwww.epfl.ch/~plaenker/BodyFitting/

Nicola D'Apuzzo
Institute of Geodesy and Photogrammetry (IGP)
Swiss Federal Institute of Technology
CH 8093 Zürich (ETHZ)
nicola@geod.ethz.ch
http://www.geod.ethz.ch/p02/projects/body/

Abstract

Synthetic modeling of human bodies and the simulation of motion is a longstanding problem in animation and much work is involved before a near-realistic performance can be achieved. At present, it takes an experienced designer a very long time to build a complete and realistic model that closely resembles a specific person. Our ultimate goal is to automate the process and to produce realistic animation models given a set of video sequences.

In this paper, we show that, given video sequences of a person moving in front of the camera, we can recover shape information and joint locations. Both of which are essential to instantiate a complete and realistic model that closely resembles a specific person and without knowledge about the position of the articulations a character cannot be animated. This is achieved with minimal human intervention. The recovered shape and motion parameters can be used to reconstruct the original movement or to allow other animation models to mimic the subject's actions.[1]

1 Introduction

Synthetic modeling of human bodies and the simulation of motion is a longstanding problem in animation and much work is involved before a near-realistic performance can be achieved. At present, it takes an experienced designer a very long time to build a complete and realistic model that closely resembles a specific person. Our ultimate goal is to automate the process and to produce realistic animation models given a set of video sequences. Eventually the whole task should be performed quickly by an operator who is not necessarily an experienced graphics designer. We should be able to invite a visitor to our laboratory, make him walk in front of a set of cameras, and produce, within a single day, a realistic animation of himself.

In this paper, we show that, given stereo video sequences of a person moving in front of the camera, we can recover shape information and joint locations, both of which are essential to instantiate the model. This is achieved with minimal human intervention: To initialize the process, the user simply clicks on the approximate location of a few key joints in one image triplet. The recovered shape and motion parameters can be used to reconstruct the original motion or to make other animation models mimic the subject's actions.

We concentrate on a video-based approach because of its comparatively low cost and good control of the dynamic nature of the process. While laser scanning technology provides a fairly good surface description of a static object from a given viewpoint, videogrammetry allows us in addition to measure and track particular points of interest, such as joints, and to record and track surface and point features around the object.

The problem to be solved is twofold: first, robustly extract image information from the data; second, fit the animation models to the extracted information. In this work, we use video sequences acquired with three synchronized cameras to extract tracking and stereo information.

Recently, techniques have been proposed [10, 7, 12, 3] to track human motions from video sequences. They are fairly effective but use very simplified models of the human body, such as ellipsoids, that do not precisely model the human shape and would not be sufficient for a truly realistic simulation.

Much work has also been devoted to the use of silhouettes for body modeling [4, 9]. They provide very useful but incomplete information about shape which is one of the issues we will address in this work. Here, we use stereo information to instantiate the sophisticated animation models that we have developed in the past to both track the mo-

It appears in the proceedings of the 1999 International Workshop on Modelling People (MPEOPLE'99), Sept. 1999, Corfu, Greece

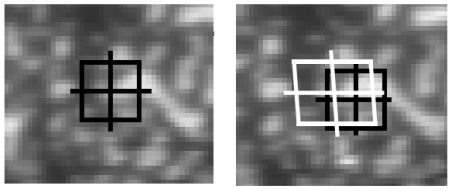


Figure 1. Least squares matching algorithm. Left: template image, right: search image

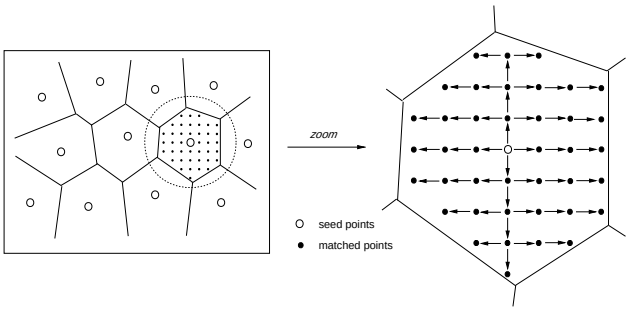


Figure 2. Search strategy for the establishment of correspondences between images

tion and recover the shape of the body as accurately as possible. However, silhouette information can easily be integrated into our extensible least-squares framework. The interested reader is referred to an earlier publication [5] for more details on silhouette integration.

We first introduce our approach to computing 3–D stereo information and 3–D surface trajectories. We then present the animation model we use. Finally, we introduce our fitting procedure and show how we can handle the different kinds of input information.

2 Extracting Image Information

2.1 Surface Measurement

Our approach is based on multi image photogrammetry. We take three images using three synchronized cameras. A multi-image least squares based matching process [8] establishes correspondences in the three images. It considers a patch of area around a selected point. One image is used as template and the others as search images. The patch in the search image is modified by an affine transformation (translation, rotation, sheering and scaling) and the grey levels are varied by multiplicative and additive constants. The algorithm finds the corresponding point in the neighborhood of the selected point in the search images by minimizing the sum of the square of the differences between the grey levels in these patches. Figure 1 shows the result of the least squares matching with an image patch of 13×13 pixels. The black boxes represent the patches selected (initial location in the search image) and the white box represents the affinely transformed patch in the search image.

To define the seed points of the multi-image matching process, approximations for a few corresponding points have to be manually selected in the three images. For example, for the arm sequence of Figure 8 we selected about ten seed points manually. The least squares algorithm is applied to find their exact location in the pictures. To define the regions between the different seed points, we compute a Voronoi tessellation in the template image. The image is divided into polyhedral regions according to which of the seed points is closest. Starting from the seed points, the stereo matcher automatically determines a dense set of correspondences. The central image is used as a template image and the other two (left and right) are used as search images. The matcher searches the corresponding points in the two search images independently. At the end of the process, the data sets are merged to become triplets of matched points.

The matcher uses the following strategy: the process starts from one seed point, shifts the template horizontally in the search image and then applies the least squares algorithm. If the quality of the match is good, the shift process continues horizontally until it reaches the region boundaries. The covering of the entire polygonal region of a seed point is achieved by subsequently horizontal and vertical shifts (Figure 2). If the quality of the match is not satisfactory, the algorithm works adaptively by changing parameters (e.g. smaller shift, bigger size of the patch). The search process is repeated for each seed point region until the whole image is covered. At the end of the process, holes of non analyzed areas can appear in the set of matched points. The algorithm tries to close these holes by searching from all directions around. The matching process results in a set of matched points in the three images. To compute the 3–D coordinates of these points, we apply forward intersection using the orientation and calibration data of the cameras [13].

2.2 Tracking process

The tracking process is also based on least squares matching techniques. The spatial correspondences between the three images of the different views and also the temporal correspondences between subsequent frames are computed using the same least squares matching algorithm mentioned as before. To start the process a triplet of corresponding points in the three images is needed. This 3–D point is then tracked through the sequence in the three images and there-

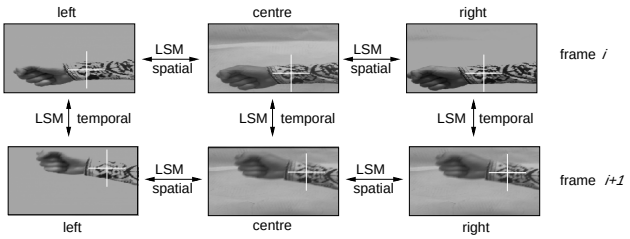


Figure 3. Tracking process

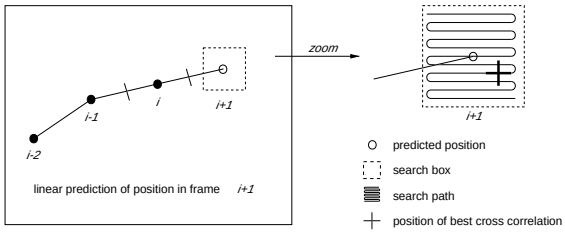


Figure 4. Tracking in image space: LSM temporal is applied at the position of best cross correlation

fore its 3–D trajectory can be computed. Figure 3 shows the use of the least squares matching algorithm to track the point.

In frame i, a triplet of corresponding points in the three images is established with the least squares matching algorithm (*spatial LSM*). In each of the three images (left, center, right) a correspondent point is matched in the next frame $i + 1$ also with the same least squares matching algorithm (*temporal LSM*). Figure 4 depicts how the temporal correspondences are established between subsequent frames.

For frame $i + 1$, a linear prediction of the position of the tracked point from the previous frame is made. A search box is defined around this predicted position in the frame $i + 1$. This box is scanned for searching the position which has the best value of cross-correlation between the image of frame i and the image of frame $i+1$. This position is considered an approximation of the exact position of the point to be tracked. The least squares matching algorithm is applied at that position and the result can be considered the exact position of the tracked point in the new frame. This process is performed independently for the three images of the different views. A *spatial LSM* is executed at the positions resulting from the *temporal LSMs* and if no significant differences occurs between the two matches, the point can be considered exactly tracked. The tracked point's 3–D trajectory is determined by computing the 3–D coordinates of the point through the sequence by forward intersection. Velocities and accelerations are also computed. The tracking algorithm is applied to all the points measured on the surface of the first frame. The result can be seen as a vector field of trajectories (position, velocity and acceleration). An important advantage of this general tracking scheme of all the points is that at the end the results can be checked for consistency and local uniformity of the movement. Two filters are applied on the results to remove or truncate false trajectories (Figure 5).

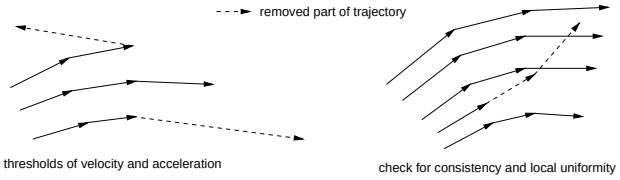


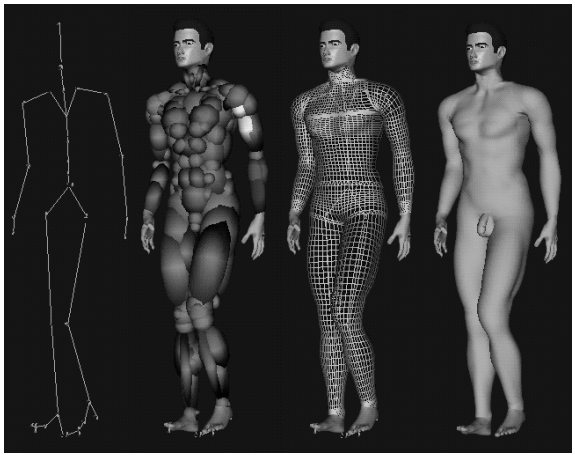
Figure 5. Vector field of trajectories of surface tracking

The first filter consists of thresholds for the velocity and acceleration. The second filter checks for the local uniformity, both in space and time, of the motion. Since the human body can be considered as an articulated moving object, the resulting vector field of trajectories must be locally uniform, i.e. the velocity vector must be nearly constant in sufficiently small regions at a particular time. To check this property, the single trajectories are compared to local (in space and time) mean values of the velocity vector. If the differences are too large, the trajectory is considered to be false and it is truncated or removed. The results of this filtering are not only trajectories without errors, but also surface measurement (in form of a 3–D points cloud) at each time instance without errors. The possible errors in the surface measurement done at the beginning of the sequence are removed during the tracking process, since they probably generate false trajectories.

Tests on different sequences have shown that the process works also with less textured surfaces like jeans or skin. Although the accuracy of the tracking is lower, the correctness of the tracked motion is assured by the uniformity filter.

3 MODELS

In this section, we first describe the complete model that we use for animation purposes. This model has too many degrees of freedom to be effectively fit to noisy data without a-priori knowledge. We therefore introduce a simplified model that we have used to derive an initial shape and position. In this work, we will use this knowledge to initialize the complete one before refining it.



(a) (b) (c) (d)

Figure 6. The layered human body model: (a) Skeleton. (b) Ellipsoidal metaballs used to simulate muscles and fat tissue. (c) Polygonal surface representation of the skin. (d) Shaded rendering.

3.1 Complete Animation Model

Generally, virtual humans bodies are structured as articulated bodies defined by a skeleton. When an animator specifies an animation sequence, he defines the motion using this skeleton.

A skeleton is a connected set of segments, corresponding to limbs and joints. A joint is the intersection of two segments, which means it is a skeleton point where the limb linked to that point may move.

Our model [17] is depicted by Figure 6. It incorporates a highly effective multi-layered approach for constructing and animating realistic human bodies. Ellipsoidal metaballs are used to simulate the gross behavior of bone, muscle, and fat tissue; they are attached to the skeleton and arranged in an anatomically-based approximation. The skin construction is made in a three step process. First, the implicit surface resulting from the combination of the metaballs influence is automatically sampled along cross-sections with a ray casting method [16, 17]. Second, the sampled points constitute control points of a B-spline patch for each body part (limbs, trunk, pelvis, neck). Third, a polygonal surface representation is constructed by tessellating those B-spline patches for seamless joining different skin pieces together and final rendering. The method, simple and intuitive, combines the advantages of implicit, parametric and polygonal surface representation, producing very realistic and robust body deformations. By applying smooth blending twice (metaball potential field blending and B-spline basis blending), the model's data size is significantly reduced.

Since the overall appearance of a human body is very much influenced by its internal muscle structures, the layered model is the most promising for realistic human animation. The key advantage of the layered methodology is that once the layered character is constructed, only the underlying skeleton need be scripted for animation; consistent yet expressive shape deformations are generated automatically.

3.2 Skeleton and State Vector

The state of the skeleton is described by the combined state vector

$$S_{body} = [S_{motion}, S_{skel}] \ . \tag{1}$$

Since the skeleton is modeled in a hierarchical manner, we can define the *static* or *init* state of the skeleton S_{skel} as the rotations and translations from each joint with respect to the preceding one. It is fixed for a given instance of the body model. The variable or *motion* state vector S_{motion} contains the actual values for each degree of freedom (DoF), i.e. the angle around the z-axis towards the next DoF. They reflect the position and posture of the body with respect to its rest position. All joints have a single angular DoF. More complicated articulations are split into several, single-DoF joints sharing the same location and only differing in their orientations.

The position of joints in a global or world referential is obtained by multiplying the local coordinates by a transformation matrix. This matrix is computed recursively by multiplying all the transformation matrices that correspond to the preceding joints in the body hierarchy:

$$X_j = \prod_i D^i(S) * X_w \ , \tag{2}$$

with $X_{j,w} = [x, y, z]^T$ being joint local, resp. world global, coordinates and the homogeneous transformation matrices D^i, which depend on the state vector S, ranging from the root articulation's first to the reference articulation's last DoF. These matrices are split into static and motion matrices, according to the state vector. They are of the form

$$D = D_{rot_z} * D_{ini} \ . \tag{3}$$

The rotation matrix D_{rot_z} is defined by the motion state vector. It is a sparse matrix allowing only a rotation around the local z-axis (Θ_κ). The static transformation $D_{ini} = (RX + sT)$ is a matrix directly taken from the standard skeleton. These matrices translate by the bone length and rotate the local coordinate system from the joint to its parent. The matrix entries are calculated using values s from the state vector S_{skel}. The variable coefficient s is necessary because the exact size of the limbs may vary from person to person.

3.3 Simplified Model of a Limb

To robustly estimate the skeleton's position and to reduce the number of DoFs, we replace the multiple metaballs of Section 3.1 by only three metaballs attached to each limb. In an earlier approach [6, 5], we used only one ellipsoid per limb. This had the advantage of being fast to compute but the errors introduced by the model's imperfection were large enough to lead to unsatisfactory fittings. We therefore decided to use a slightly more complicated model which approximates better the shape of human limbs. Figure 7 shows the model we have used to recover the shape and motion from the arm sequences of Figure 8. To reduce the number of DoFs we introduced higher level parameters which cover a number of direct metaball parameters. Like "upper arm width" which controls the relative size of all metaballs in the region of the upper arm.

The metaballs are rigidly attached to the skeleton. They have a fixed orientation and a fixed position relative to the length of the limb. Only their size, i.e. their radii, are subject to modification by the fitting process.

The different body parts are segmented before the fitting starts. This is simply done during the initialization phase where the model takes an approximate posture which is good enough to assign a 3–D observation to the closest limb. Thus, we do not have to wait for a motion of the person to split a limb such as the arm into two parts, upper arm and forearm, as is the case in the work of Kakadiaris and Metaxas [11]. The segmentation is reversible as it is redone after several iterations and, thus, possible segmentation errors due to a wrong initialization are removed during the fitting process.

More sophisticated primitives that include both global and local deformations, such as tapered superquadrics [11] or evolving surfaces [14], may be able to approximate more closely the exact shape of the limb. However, they require the setting of more parameters and are thus harder to fit. As noted in Section 3.1, the double-blending approach provides realistic looking shapes by using only few and simple primitives.

3.4 Metaballs and their Mathematical Description

3.4.1 Definition

In Blinn's basic formulation [2], metaballs or *blobs* are defined by a set of points $P_i(x_i, y_i, z_i)$ that are the sources of a potential field. Each source is defined by a *field function* $F_i(x, y, z)$ that maps $\mathbb{R}^3$ to $\mathbb{R}$, or a subset of $\mathbb{R}$. At a given point $P(x, y, z)$ of the Euclidean space, the fields of all sources are computed and added together, leading to the global field function $F(x, y, z) = \sum_{i=1}^{n} F_i(x, y, z)$. A curved surface can then be defined from the global field function F by giving a threshold value T and rendering the following equipotential surface $\mathbf{S}$ for this threshold:

$$\mathbf{S} = \left\{(x, y, z) \in \mathbb{R}^3 \mid F(x, y, z) = T\right\} \quad . \tag{4}$$

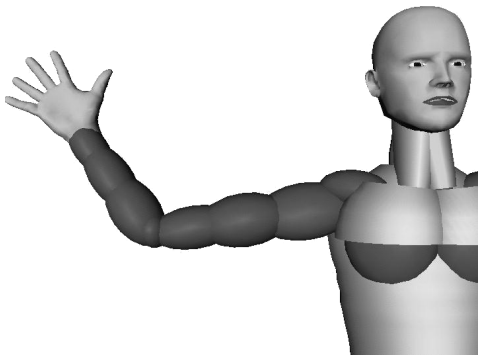


Figure 7. Simplified model for fitting. Although the metaballs are displayed as distinct ellipsoids, they blend into each other to form a single smooth surface.

Conceptually it is usually simpler to consider field function F_i as the composition of two functions [1]: the *distance function* d_i which maps $\mathbb{R}^3$ to $\mathbb{R}^+$, and the *potential function* f_i which maps $\mathbb{R}^+$ to $\mathbb{R}$:

$$F(x, y, z) = \sum_{i=1}^{n} f_i(d_i(x, y, z)) \quad . \tag{5}$$

The function $f_i(d)$ characterizes the distance between a given point $P(x, y, z)$ and the source point $P_i(x_i, y_i, z_i)$. Typically d_i is defined as a function of a user-provided parameter $r_a \in \mathbb{R}^+$ (called *effective radius*) which expresses the growing speed of the distance function. The most obvious solution for $d_i(x, y, z)$ is the Euclidean distance, but several other functions have been proposed in the literature, especially when the potential source is not reduced to a single point or its field is not equally distributed in space.

3.4.2 Distance function

In this work, we only consider ellipsoids as primitives because they are relatively simple but, nevertheless, allow modeling of human limbs with a fairly low number of primitives and thus number of parameters. We represent the distance function d_i by the implicit distance to the ellipsoid that is

$$d_i(x, y, z) = \left(\frac{x}{l_x}\right)^2 + \left(\frac{y}{l_y}\right)^2 + \left(\frac{z}{l_z}\right)^2 \quad , \tag{6}$$

where $L_i = (l_x, l_y, l_z)$ are the radii of the ellipsoid, i.e. half the axis length along the principal directions.

3.4.3 Potential function

The field value at any point P in space is defined by the distances between P and the source points P_i. The center of the primitive, its source, has the greatest density. The value of the primitive's density, or *weight*, decreases toward the element's outer edge, or effective radius. The visible size of a primitive, called the *threshold radius*, is determined by the effective radius and weight. Field functions should satisfy two criteria:

1. Extremum: The contribution at the source is some maximum value w_0, and the field will drop smoothly to zero at a distance r_a, the effective radius.

2. Smoothness: In order to blend multiple metaballs smoothly and gradually, $f'(0) = f'(r_a) = 0$.

A single, lower degree polynomial cannot meet both criteria, hence either piecewise quadric or high order polynomials have been proposed. Their disadvantage are a high complexity and thus high computational cost.

Here we are attempting to fit the model to 3–D data by minimizing an objective function. In order to do so, we need to work on a well-defined mathematical basis and the smoothness criterion is essential when fitting a shape with multiple metaballs. We therefore use an exponential field function:

$$f_i = w_i \left(\frac{1}{e^d}\right)^2 = w_i * exp(-2d) \ , \tag{7}$$

with d being defined as in Equation 6 and the weight being fixed for the moment ($w_0 = 1$, $w_t = 0.5$). In the future, we might leave the weight as a free parameter for the fitting since it allows to easily model sharper edges.

An exponential field function is also more effective in the least squares fitting framework because its derivatives are very easy to compute. Its equipotential surface $\mathbf{S}$ is only slightly different from the standard representation and, more importantly, it never falls to zero.

This last property has two consequences:

1. Each blob has an influence on all other blobs of the same limb, although, it will become very small for distant blobs. This is obviously undesired for modeling purposes since the designer looses local control.

2. At the same time as each blob influences all other blobs, each blob is influenced by all observations in our fitting framework. This allows us to work with only a rough initialization of the model's posture because of the long range effect of the $exp()$ function. Since the observations are already segmented and associated to body parts, the unlimited influence does not pose any problems on the other body parts.

4 Fitting the Models to Image Data

From a fitting point of view, the body model of Section 3.3 embodies a rough knowledge about the shape of the body and can be used to constrain the search space. Our goal is to fix its degrees of freedom so that it conforms as faithfully as possible to the image data.

Here we use motion sequences such as the one shown in Figure 8 and corresponding stereo data computed using the method of Section 2.1. Thus, the expected output of our system is a state vector that describes the shape of the metaballs and a set of joint angles corresponding to their positions in each frame.

In this section, we introduce the least squares framework we use and show how we can exploit the tracking and stereo data that we derive from the images.

4.1 Least Squares Framework

In standard least-squares fashion, we will use the image data to write $nobs$ observation equations of the form

$$f_i(S) = obs_i - \epsilon_i \ \ , 1 \leq i \leq nobs \ \ , \tag{8}$$

where S is the state vector of Equation 1 that defines the shape and position of the limb and ϵ_i is the deviation from the model. We will then minimize

$$v^T P v \Rightarrow Min \ , \tag{9}$$

where v is the vector of residuals and P is a weight matrix associated with the observations (P is usually introduced as diagonal).

Our system must be able to deal with observations coming from different sources that may not be commensurate with each other. Formally we can rewrite the observation equations of Equation 8 as

$$f_i^{type}(S) = obs_i^{type} - \epsilon_i \ \ , 1 \leq i \leq nobs \ \ , \tag{10}$$

with weight p_i^{type}, where $type$ is one of the possible types of observations we use. In this paper, $type$ is restricted to object space coordinates, although other information cues can easily be integrated.

The individual weights of the different types of observations have to be homogenized prior to estimation according to:

$$\frac{p_i^k}{p_j^l} = \frac{\left(\sigma_j^l\right)^2}{\left(\sigma_i^k\right)^2} \ , \tag{11}$$

where σ_j^l, σ_i^k are the a priori standard deviations of the observations obs_i, obs_j of type k, l.

Applying least-squares estimation implies the joint minimum

$$\sum_{type=1}^{nt} v^{type} P_{type} v^{type} \Rightarrow Min \ , \tag{12}$$

with nt the number of observation types, which then leads to the well-known normal equations which need to be solved using standard techniques.

Since our overall problem is non-linear, the results are obtained through an iteration process. We use a modified version of the Levenberg-Marquardt algorithm from [15] which is able to deal with the huge number of observations we encounter.

4.2 Using Tracking Data

The 3–D tracking information of Section 2.2 serves to capture robust stereo information and to initialize the body model in all frames. The algorithm is initialized by letting the user specify an approximate posture and position of the model in the first frame of the sequence. The results of the tracking deliver the approximate positions of the visible articulations for the rest of the sequence.

4.3 Using Stereo Data

3–D points such as the ones computed with the technique of Section 2.1 or any other source of 3–D information can be used. We want to minimize the distance of the reconstructed limb to all such "attractor" points. Given the implicit description of our metaballs, the simplest way to achieve this result is to write a pseudo-observation equation of the form:

$$\sum_{i=1}^{np} w_i \cdot \left(\frac{1}{e^{d_i}} \right)^2 = w_t - \epsilon \qquad (13)$$

$$\sum_{i=1}^{np} \left(\frac{1}{e^{\left(\frac{x_i}{lx_i}\right)^2 + \left(\frac{y_i}{ly_i}\right)^2 + \left(\frac{z_i}{lz_i}\right)^2}} \right)^2 = \frac{1}{2} - \epsilon \ , \qquad (14)$$

where np is the number of primitives for this body part, $P_i\,(x, y, z)$ is the 3–D observation transformed into the local coordinates of primitive i with radii $L_i(lx, ly, lz)$. We use Equation 14 which is the same than Equation 13 except for the fixed weights $w_t = \frac{1}{2}, w_i = 1,\ i \in [1, np]$.

The optimization is effected wrt. the primitives' radii L_i and the DoFs which reside in the transformation of each observation from world global to primitive local coordinates. These DoFs consist of the motion parameters and the skeleton parameters, i.e. length of each limb. According to Equation 2, each P_i can be written as a function of its world coordinates and the elements of state Vector S. In practice, we experienced better convergence by iteratively alternating between primitive parameters and skeleton parameters instead of optimizing them simultaneously. For more detail we refer the interested reader to a previous publication [5].

Figure 8. Arm sequence used to test the algorithms.

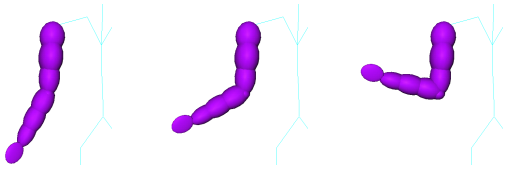


Figure 9. Simplified arm model after being fitted to the 3–D information obtained from our stereo algorithm of the images of Figure 8. Shoulder angle differs because we don't get any information about the arm's posture wrt. the body.

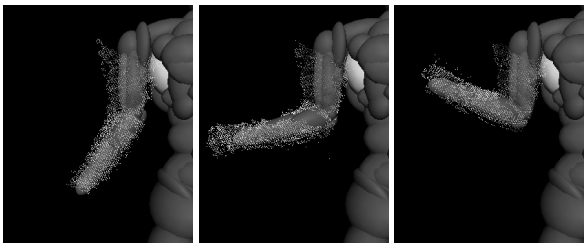


Figure 10. Full arm model after being fitted. Note the short upper arm which is "correct" in the sense that the original images are cropped too far from the shoulder.

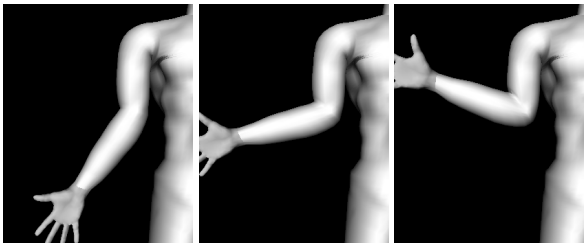


Figure 11. The recovered shape and animation parameters applied to the full animation model.

4.4 Preliminary Results

Figure 8 shows three frames from a sequence of a person waving his arm in front of the cameras. After having computed 3–D stereo data from these images we first fit the simplified arm model to the data (fig. 9). In a second phase, we fit our full model of the right arm and we are able to reconstruct the positions and shapes depicted by Figure 10. Since we use the layered model approach, compare Section 3.1, a skin can be computed automatically on top of the metaballs. Figure 11 shows the "ready for production" model whose dimensions closely resemble the filmed person.

5 Conclusion and Future Work

In this paper, we have shown that given video sequences of a moving person acquired with a multi-camera system, we can recover shape information and track joint locations during the motion. We have outlined techniques for fitting a complete animation model to noisy stereo data and we have presented a new tracking process based on least squares matching. The recovered shape and motion parameters can be used to create a realistic animation. Our ultimate goal is to produce automatically, with minimal human intervention, realistic animation models given a set of video sequences. The capability we intend to develop will be of great applicability in animation areas, since the techniques used nowadays require a very long time of manual work to generate and animate sophisticated models of humans. Automating the process will allow an increase of realism with simultaneous decrease of costs.

In future work, we will next produce some synthetic data in order to test the accuracy of the system. At the moment we are applying the algorithms on sequences of full bodies in motion. We will also investigate the possibilities of having the model guide the tracking process. If a point on the body's surface vanishes due to occlusion we can employ the model to predict where and when it will appear again.

6 Acknowledgments

The work reported here was funded in part by the Swiss National Science Foundation.

References

[1] C. Blanc and C. Schlick. Extended field functions for soft objects. In *Eurographics Workshop on Implicit Surface 95*, pages 21–32, Grenoble, France, 1995.

[2] J. F. Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics*, 1(3):235–256, 1982.

[3] C. Bregler and J. Malik. Tracking people with twists and exponential maps. In *Conference on Computer Vision and Pattern Recognition*, Santa Barbara, CA, June 1998.

[4] J. Davis and A. Bobick. A Robust Human-Silhouette Extraction Technique for Interactive Virtual Environments. In *Workshop on Modelling and Motion Capture Techniques for Virtual Environments*, pages 12–25, Geneva, Switzerland, November 1998.

[5] P. Fua, A. Gruen, R. Plänkers, N. D'Apuzzo, and D. Thalmann. Human body modeling and motion analysis from video sequences. In *International Archives of Photogrammetry and Remote Sensing*, volume 32(B5), pages 866–873, Hakodate, Japan, 1998.

[6] P. Fua, R. Plänkers, and D. Thalmann. From image synthesis to image analysis: Using human animation models to guide feature extraction. In *Fifth International Symposium on the 3-D Analysis of Human Movement*, Chattanooga, TN, July 1998.

[7] Gavrila and L. Davis. 3d model-based tracking of humans in action : A multi-view approach. In *Conference on Computer Vision and Pattern Recognition*, pages 73–80, San Francisco, CA, June 1996.

[8] A. Gruen. Adaptive least squares correlation: a powerful image matching technique. *South African Journal of Photogrammetry, Remote Sensing and Cartography*, 14(3):175–187, 1985.

[9] A. Hilton, D. Beresford, T. Gentils, R. Smith, and W. Sun. Virtual People: Capturing Human Models to Populate Virtual Worlds. In *Computer Animation*, Geneva, Switzerland, May 1999.

[10] Kakadiaris and Metaxas. Model based estimation of 3d human motion with occlusion based on active multi-viewpoint selection. In *Conference on Computer Vision and Pattern Recognition*, pages 81–87, San Francisco, CA, June 1996.

[11] I. Kakadiaris, D. Metaxas, and R. Bajcsy. Active part-decomposition, shape and motion estimation of articulated objects: A physics-based approach. In *Conference on Computer Vision and Pattern Recognition*, pages 980–984, 1994.

[12] F. Lerasle, G. Rives, M. Dhome, and A. Yassine. Human Body Tracking by Monocular Vision. In *European Conference on Computer Vision*, pages 518–527, Cambridge, England, April 1996.

[13] H.-G. Maas. Image sequence based automatic multi-camera system calibration techniques. In *International Archives of Photogrammetry and Remote Sensing*, volume 32(B5), pages 763–768, Hakodate, Japan, 1998.

[14] R. Malladi, J. Sethian, and B. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[15] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes, the Art of Scientific Computing*. Cambridge U. Press, Cambridge, MA, 1986.

[16] J. Shen and D. Thalmann. Interactive shape design using metaballs and splines. In *Implicit Surfaces*, April 1995.

[17] D. Thalmann, J.Shen, and E. Chauvineau. Fast Realistic Human Body Deformations for Animation and VR Applications. In *Computer Graphics International*, Pohang,Korea, June 1996.