Human Motion Tracking by Combining View-Based and Model-Based Methods for Monocular Video Sequences*

Jihun Park, Sangho Park, and J.K. Aggarwal

[1] Department of Computer Engineering
Hongik University
Seoul, Korea
jhpark@hongik.ac.kr
[2] Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
sangho@ece.utexas.edu
[3] Department of Electrical and Computer Engineering
The University of Texas at Austin
Austin, TX 78712
aggarwaljk@mail.utexas.edu

Abstract. Reliable tracking of moving humans is essential to motion estimation, video surveillance and human-computer interface. This paper presents a new approach to human motion tracking that combines view-based and model-based techniques. Monocular color video is processed at both pixel level and object level. At the pixel level, a Gaussian mixture model is used to train and classify individual pixel colors. At the object level, a 3D human body model projected on a 2D image plane is used to fit the image data. Our method does not use inverse kinematics due to the singularity problem. While many others use stochastic sampling for model-based motion tracking, our method is purely dependent on parameter optimization. We convert the human motion tracking problem into a parameter optimization problem. A cost function for parameter optimization is used to estimate the degree of the overlapping between the foreground input image silhouette and a projected 3D model body silhouette. The overlapping is computed using computational geometry by converting a set of pixels from the image domain to a polygon in the real projection plane domain. Our method is used to recognize various human motions. Motion tracking results from video sequences are very encouraging.

* This work was partially supported by grant No. 2000-2-30400-011-1 from the Korea Science and Engineering Foundation. We thank Ms. Debi Prather for proofreading of this paper.

V. Kumar et al. (Eds.): ICCSA 2003, LNCS 2669, pp. 650–659, 2003.

1 Introduction and Related Work

Reliable tracking of moving humans is essential to motion estimation, video surveillance and human-computer interface. Tracking non-rigid objects such as moving humans presents several difficulties for computer analysis. Problems include segmentation of the human body into meaningful body parts, handling occlusion, and tracking the body parts along the image sequence. The approaches that have been proposed for tracking a human body can be classified into two groups: model-based approaches and view-based approaches. Refer [1,6] for reviews. Model-based approaches use *a priori* models explicitly defined in terms of kinematics and dynamics. View-based approaches use heuristic assumptions when no *a priori* model is available. These two approaches can be combined at various levels to increase efficiency [15]. This paper presents a new approach to human motion tracking that combines techniques from both view-based and model-based approaches. The proposed system processes the input image sequence at both pixel level and semantic object level. At the pixel level, a Gaussian mixture model is used to classify individual pixels into several color classes, which are merged into coherent blobs by relaxation labeling. At the object level, a 3D human body model projected to the 2D image plane is used to fit the image data. The view-based processing at the pixel level efficiently reduces the overhead in model-based processing at the object level by providing foreground silhouettes.

All kinematics-based motion tracking methods may be classified into two groups depending the use of inverse kinematics – computing joint angles given end-tip kinematic parameters. If no inverse kinematics is used, we call that approach a *forward kinematics-based approach*; otherwise we call it an *inverse kinematics-based approach*. Our work is forward kinematics-based, while the papers reviewed here [8,12] are inverse kinematics-based.

Morris and Rehg [12] presented one of the first model-based methods for deriving differential inverse kinematics equations for image overlapping, although differential kinematics originated from robotics [5]. Morris and Rehg [12] used a 2D scaled prismatic model for figure fitting, and reduced the singularity problem by working in the 2D projection plane. But singularity is inevitable because this method is based on differential inverse kinematics. Huang, et al., [8] extended the inverse kinematics work presented in [12] to solve motion parameters of the articulated body in a statistical framework using the expectation-maximization (EM) algorithm. Sidenbladh et al. [17] converted the human motion tracking problem into a probabilistic inference problem aimed at estimating the posterior probability of body model parameters given an input image.

The rest of the paper is organized as follows: Section 2 describes the procedure at the pixel level and describes the blob formation. Section 3 presents the human body modeling and forward kinematics-based cost function for fitting. Section 4 describes the issue of kinematics and singularity, explaining why we use a forward kinematics-based approach to avoid the singularity problem. Results and conclusions follow in Section 5.

2 Pixel Classification

2.1 Color Representation and Background Subtraction

Most color cameras provide an RGB (red, green, blue) signal. The RGB color space is, however, not effective for human visual perception of color and brightness. Here, the RGB color space is transformed to the HSV (hue, saturation, value) color space to make the intensity or brightness explicit and independent of the chromaticity.

Background subtraction is performed in each frame to segment the foreground image region. Refer to [13] for details. The color distribution of each pixel $v(x, y)$ at image coordinate (x, y) is modeled as a Gaussian. Using k_b training frames ($k_b = 20$), the mean $\mu(x, y)$ and standard deviation $\sigma(x, y)$ of each color channel is calculated at every pixel location (x, y). Foreground segregation is performed for every pixel $v(x, y)$, by using a simple background model, as follows: at each image pixel (x, y) of a given input frame, the change in pixel intensity is evaluated by computing the Mahalanobis distance from the Gaussian background model.

$$\delta = \frac{|v(x, y) - \mu(x, y)|}{\sigma(x, y)} \tag{1}$$

The foreground image $F(x, y)$ is defined by the maximum of the three distance measures, δ_H, δ_H, and δ_V for the H, S, V channels;

$$F(x, y) = max[\delta_H(x, y), \delta_S(x, y), \delta_V(x, y)] \tag{2}$$

F is then thresholded to make a binary mask image. At this stage, morphological operations are performed as a post-processing step to remove small regions of noise pixels.

2.2 Gaussian Mixture Modeling for Color Distribution

In HSV space, the color values of a pixel at location (x, y) are represented by a random variable $v = [H, S, V]^t$ with the vector dimension $d = 3$. According to the method in [13], color distribution of a foreground pixel v is modeled as a mixture of C_0 Gaussians weighted by prior probability $P(\omega_r)$, given by;

$$p(v) = \Sigma_{r=1}^{C_0} p(v|\omega_r) P(\omega_r) \tag{3}$$

where the r-th conditional probability is assumed as a Gaussian, as follows:

$$p(v|\omega_r) = (2\pi)^{-d/2} |\Sigma|^{-1/2} exp[-\frac{(v - \mu_r)^t \Sigma^{-1} (v - \mu_r)}{2}], r = 1, ..., C_0 \tag{4}$$

Each Gaussian component is represented by the prior probability $P(\omega_r)$ of the r-th color class ω_r, a mean vector μ_r of the pixel color component, and a covariance matrix Σ_r of the color components.

2.3 Training the Gaussian Parameters

In order to obtain the Gaussian parameters, an EM algorithm is used with the first η frames of the sequence as the training data ($\eta = 5$). Initialization (E-step) of the Gaussian parameters is done as follows: all prior probabilities are assumed as equal, i.e., the mean is randomly chosen from a uniform distribution within a possible pixel value range, and the covariance matrix is assumed to be an identity matrix. Training (M-step) is performed by iteratively updating the above-mentioned parameters. Refer [4] for details. The iteration stops if either the change in the value of the means is less than 1 percent with respect to the previous iteration or when a user-specified maximum iteration number is exceeded. We start with 10 Gaussian components ($C_0 = 10$) and merge similar Gaussians after the training by the method in [10], resulting in C Gaussians. The parameters of the established C Gaussians are then used to classify pixels into one of the C classes in subsequent frames.

2.4 Classification of Individual Pixels

The color classification of the individual pixels is achieved by a maximum *a posteriori* (MAP) classifier. Once the Gaussian mixture model G for pixel color is obtained, we compute the MAP probability that each pixel in the subsequent frames belongs to each Gaussian component. The class that produces the largest probability value for a pixel v is chosen as the pixel-color class label ω_L for that pixel.

$$\omega_L = argmax_r(log(P(\omega_r|v))), 1 \leq r \leq C \tag{5}$$

2.5 Relaxation Labeling

The pixel color classification results in free-form blobs of different color labels adjacent to one another. Connected component analysis is used to register adjacent blobs. Relaxation labeling [16,13] is performed to remove small blobs and to achieve coherent large blobs according to the color similarity of the adjacent blobs, as follows: two adjacent blobs A_i and A_j are merged together if they are similar in color, where the similarity is defined by the Mahalanobis distance δ_Φ of color feature Φ between A_i and A_j, as follows:

$$\delta_\Phi = (\Phi_i - \Phi_j)^T {\Sigma_\Phi}^{-1} (\Phi_i - \Phi_j) \tag{6}$$

$$\Phi = [\mu_H, \mu_S, \mu_V]^T \tag{7}$$

where Σ_Φ is the covariance matrix of color channels for all blobs in the image. Blobs A_i and A_j are merged if δ_Φ is less than the threshold T_Φ, which was obtained from training data. (Refer [13] for details.) The result of the relaxation labeling is a set of blobs that segment the foreground image into coherent regions based on color and position, which provides view-based initial segmentation and tracking of the foreground human figures. More detailed segmentation and tracking is achieved at the object level that incorporate a 3D human body model projected onto the 2D image plane.

3 Human Body Modeling and Cost Function

3.1 Human Body Modeling

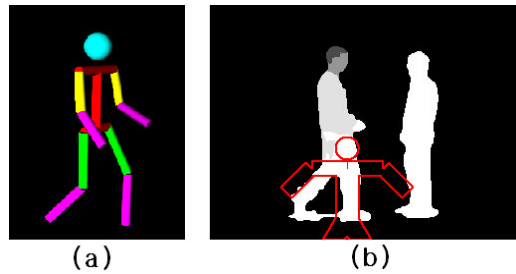

Fig. 1. 3D Model body(a), and overlapping between image and projected model body(b).

As shown in figure 1(a), the body is modeled as a configuration of nine cylinders and one sphere. (Two extra horizontal cylinders are drawn for better illustration, but do not exist in our modeling.) These are projected onto a 2D real projection plane. A sphere represents the head, while the rest of model body is modeled using cylinders of various radii and lengths. Our model body is very similar to that used in [17]. We may use more sophisticated tapered/oval-circular cylinders, but our method is robust enough that a rough body modeling is enough to track foreground images. Currently, we use only nine 1-DOF (1 degree-of-freedom) joints plus body displacement DOF, and a vertical rotation to compensate for a camera view. These are our control variables for the cost function. Body parts are linked together by kinematic constraints in a hierarchical manner. This can be considered to be a tree structure with the base at the pelvis (the bottom of a cylinder represents the torso) of the model body. The methods presented in [8] cannot handle body occlusion, and ignore half of the body. We have overcome the body occlusion problem using computational geometry, by computing the union of the projected model body parts and then computing the intersection with overlapping input image silhouettes.

Due to possible variations in the size of human subjects, our method must be able to handle various magnitudes of height and thickness of the human in input images. We can handle this problem by initially counting the number of pixels from a pixel density map of a preprocessed image to guess the height of a body. Then, we can run an initial parameter optimization, solving by making the thickness of a body an additional variable while using a pre-guessed human height. We assume that humans are standard in their body segment length ratio. After solving the best thickness for a model body, the thickness variable becomes a constant and remains set as a constant for the remaining body tracking over various input images for the same human model. Figure 1(b) shows the initial state of searching for the best overlapping configuration.

3.2 Forward Kinematics-Based Cost Function

The kinematics problem is well explained in Craig's book [3]. A projected model body is assumed to have an affine transformation [8]. This is mathematically acceptable if an orthographic camera model is used and the model body moves parallel to the projection plane. Given a set of joint angles and body displacement values, the forward kinematics function, $h(\cdot)$, where $\cdot$ is a generic variable(s), computes points of body segment boundaries. The P matrix then projects the computed boundary points on a 2D projection plane. The projected boundary points are compared to the foreground image silhouette without using color information from the foreground image to match the image silhouette. $g(\cdot)$ converts projected points to a polygon(s). The input image is preprocessed using $f(\cdot)$. The projection plane is represented in the real numbers. $r(\cdot)$ converts the input image silhouette to a polygon(s) in real number domain. The representation in the real number domain makes the derivative-based parameter optimization possible. For easier tracking of the body, we use head part information from image preprocessing results [13]. The distance between the silhouette head center and the model head center is minimized. However, we can track body motion without head information using the following equation:

$$\begin{aligned} c(I,\theta) = & -w_1 \times [a(r(f(I)) \cap (\cup_l g(P \cdot h_l(\theta,t))))] \\ & + w_2 \times \sum_{xy} (w_d(x,y) \times a(d(x,y) \cap (\cup_l g(P \cdot h_l(\theta,t))))) \\ & + w_3 \times (h_{hc}(\theta,t) - q(f(I)))^2 \end{aligned} \tag{8}$$

Let us explain the notation used in equation 8 in more detail. P is an orthographic camera projection matrix, projecting a 3D model body to the 2D plane. This is a constant matrix, assuming that camera parameters are fixed. $h_l(\cdot)$ is a nonlinear forward kinematics function of an *l-th* body part in terms of joint DOF. $\bar{\theta}$ is a joint DOF vector, represented in a column matrix. (We denote $\bar{x}$ as vector x.) $\bar{\theta}$ is a function of time when a complete sequence is considered. $g(\cdot)$ is a function that takes 2D input points and converts them to a polygon. $r(\cdot)$ is a real function that takes an image as an input and converts its foreground (non-zero value) part to a set of polygons, possibly with holes. Given a raw image, $f(\cdot)$ returns a preprocessed image. Given a preprocessed human image, $q(\cdot)$ computes a head center. I is a raw input image. $I(x,y)$ denotes a grey level pixel value at image location (x,y). $d(x,y)$ is a square polygon of area size 1, representing a pixel located at the (x,y)-*th* position in the distance map [6]. t means time related with frames. $w_d(x,y)$ is a distance map value of $(b(I))(x,y)$, and has a scalar value, where $b(I)$ is background part of input image. $\cap$ is an operation that takes two polygons and returns an intersected polygon(s). $\cup$ is an operation that takes two polygons and returns a unioned polygon(s). $a(\cdot)$ is a function that takes a polygon and returns the size of its area. $c(\cdot)$ is a cost function for parameter optimization that depends on a raw input image I and model DOF variables θ. $h_{hc}(\cdot)$ is the model head center function using forward kinematics, and w_i, $i = 1, 2, 3$ are weighting factors.

Because the input is $\bar{\theta}$, a vector of joint DOF values, this computation is purely forward kinematics-based and thus presents no singularity problem. In deriving forward kinematics equations, we set local coordinates for various parts of the body to make a hierarchical body. We can assign joint limits for model body parts. Only a model-based approach allows easy geometric constraint computation, which is implemented as possible ranges for each variable. Head part information is provided from image preprocessing. Our method is robust enough, regardless of head tracking, that we can find a best match from image silhouette matching. However, the weakness of our approach is in estimating how many humans need to be tracked.

At the start, the model body has initially guessed arbitrary input values. Even though there is no overlapping between the projected model body and the image silhouette, the cost function knows which direction is the best move to minimize the cost function, due to the distance map. The value of our cost function decreases when by moving the projected model body is moved to the foreground image, because overlapping of the foreground image increases while overlapping with background decreases. The parameter optimizer will find the best fitting configuration that produces the optimum value of the cost function. If there is an occlusion between two persons, our method must rely on a distance map [6] for each person to track correctly.

3.3 Computational Geometry for the Cost Function

An image silhouette is one that is converted from the 2D integer pixel domain to a real domain such that the resulting image silhouette becomes a jagged-edge polygon with only horizontal and vertical edges. Each body part is projected on the 2D projection plane. In this process, occlusion occurs among body parts. Basically, projected objects are either polygon shaped or circlular. Because we are concerned with how well the model body overlaps with the foreground image silhouette, it is quite natural that the resulting body parts occlude each other. The projected occluded body parts are unioned using a modified version of Weiler-Atherton's polygon union algorithm [7]. The resulting polygon-like object(s) may even have holes in it, showing our computational geometry works fine. We compute the polygon intersection between the image silhouette and the model silhouette. We use pixel-based computational geometry because it will allow us to compute a cost function based on a distance map, which has different values for each pixel. We found the best fitting configuration using the GRG2 [11] optimization package with the cost function discussed so far.

4 Kinematics and Singularity

We avoid using inverse kinematics due to its singularity problem. We may also work in differential kinematics. In terms of differential kinematics, we usually work either on the velocity level or on the acceleration level. Working on highly differentiated equations does not always have merits. Computed velocities may

have errors in image overlapping computation, which may make our differential kinematics computation unreliable. Many papers presented up to now are based on inverse kinematics and usually work on the velocity level. This type of approach requires computation of joint angular velocities given the end tip velocities. This process involves a matrix, called a Jacobian inversion. If the Jacobian matrix is not square, one must compute a pseudo-inverse Jacobian matrix [2]. Even if the matrix is square, it can be singular. Because they take error-prone difference values computed between two images, the resulting computation is not truly reliable [12]. Our approach, on the other hand, computes in a pure forward kinematics-based way. This is the major difference from others' work. Let $\bar{f}(\bar{\theta})$ be our forward kinematics equation where $\bar{\theta}$ is a body DOF-related vector consisting of joint angles and body displacements. In our method, we control $\bar{\theta}$. Given $\bar{\theta}$, we get the resulting $\bar{f}$ values. If the $\bar{f}$ values are not the values we need, we modify the input $\bar{\theta}$. This process is done by parameter optimization. After this computation, we interpolate $\bar{\theta}$ to get approximate $\dot{\bar{\theta}}$ and $\ddot{\bar{\theta}}$. Differential kinematics-based methods, on the other hand, use differential kinematics equations. By differentiating $\bar{f}(\bar{\theta})$, they can get $\dot{\bar{f}} = \frac{\partial \bar{f}}{\partial \bar{\theta}} \dot{\bar{\theta}}$. $\frac{\partial \bar{f}}{\partial \bar{\theta}}$ is called a Jacobian matrix. They get $\dot{\bar{f}}$ values from input images. In order to compute $\dot{\bar{\theta}}$, they need to compute $\frac{\partial \bar{f}}{\partial \bar{\theta}}^{-1}$. But if this matrix is not square, they must compute a pseudo-inverse matrix [2]. However, even if the matrix is square, there is some chance that it is singular. For singular matrix inversion, they usually use singular value decomposition [14].

5 Experimental Results and Conclusion

We have worked on several different 2D-based human motions. The example shown in figure 2 has a static background with one person (the left) pushing another (right) person away. The white line shows the result of union of every model body part. There are even holes in the resulting body, polygon-like object. We can track every person appearing in the scene, and as long as there is no heavy occlusion between the subjects, motion tracking is satisfactory. Using body part information, our method can handle even heavy occlusion to a limited degree, although the tracking quality is degraded. The graph of figure 2 shows the corresponding pair of joint angles of torso, shoulder, and elbow of the pushing motion of the model body as well as the real human in the scene. All joint angle data on the graph is for the left arm. We measured human joint angles from images and compared it with model tracking data. In the graph labels, **M** means model body data while **R** means real body data. **U** means upper while **L** means lower. Excellent matching was achieved for the Model left upper arm (**M L U ARM**) data to the real left upper arm (**R L U ARM**) data as well as for model torso data (**M TORSO**) to real torso data (**R TORSO**). In order to compute model data, we did individual motion configuration fitting for every frame. Only the left arm of the model body is tracked for the pushing motion. This is quite natural and is the current limitation of using monocular video input, which provides very limited input information. Our method finds

one locally optimum solution from the search space, as can be seen in the graphs, and the tracking is good. In the middle of the pushing motion, there is a body occlusion between the two persons. Because our cost function is pixel-area based, in the third image of figure 2, it appears that the left person's hand is on the right person's face. This is due to not using body part information for tracking motion, and can be fixed if body part information is used. Figure 3 shows the walking (approaching) and shaking hands motions.

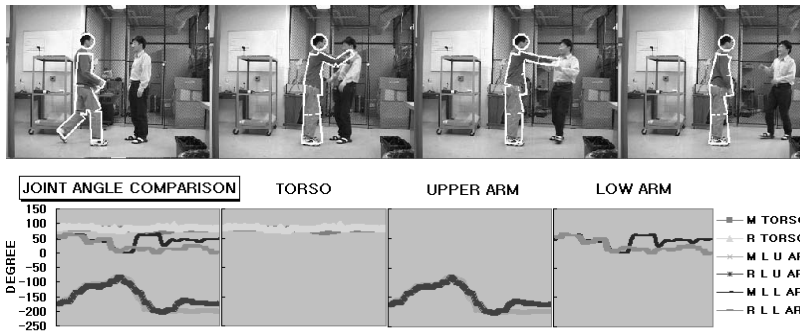

Fig. 2. The subject with the model figure superimposed, shown over a pushing motion. Corresponding joint angle graph of torso, shoulder, and elbow.

In this paper, we presented a new approach to human motion tracking that combines view-based and model-based methods at the pixel level and the object level, respectively. The view based-methods at the pixel level use a Gaussian mixture model and a relaxation labeling technique to achieve initial segmentation of the foreground human figures. The initial segmentation is data-driven, and it significantly reduces the overhead in model initialization and fitting. The model based-method at the object level uses a 3D human body model and parameter optimization techniques to achieve refined segmentation and tracking of the moving humans. Using the human body model achieves robustness of our system. Our forward kinematics-based system overcomes the problem of singularity in inverse kinematics-based systems, and our nonlinear optimization-based fitting does not depend on the number of particles as is the case in stochastic sampling-based approaches. We have presented a solution to the model body part occlusion problem [8] using computational geometry. As demonstrated in figure 3, the motion tracking results from video sequences are very encouraging.

References

1. J.K. Aggarwal and Q. Cai. Human motion analysis: a review. *Computer Vision and Image Understanding*, 73(3):295–304, 1999.
2. H. Asada and J. Slotin. *Robot Analysis and Control*. John Wiley and Sons, New York, NY, 1985.

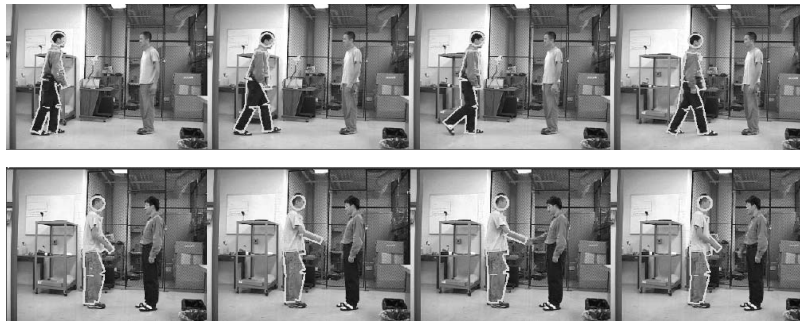

Fig. 3. The subject, with the model figure superimposed, shown over a walking (approaching) motion, and a shaking hands motion.

3. J. Craig. *Introduction to Robotics Mechanics and Control*. Addison-Wesley, Reading, MA, 1986.
4. R.O. Duda, P. Hart, and E. Stork. *Pattern Classisifcation*, chapter Unsupervised Learning and Clustering, pages 517–583. Wiley, New York, 2 edition, 2001.
5. R. Freeman and D. Tesar. Dynamic Modeling of Serial and Parallel Mechanisms / Robotic Systems : Part I - Methodology. In *Trends and Developments in Mechanisms, Machines and Robotics, 20th Biennial Mechanisms Conference*, 1988.
6. D. Gavrila. The visual analysis of human movement: a survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.
7. F. Hill. *Computer Graphics*. Macmillan, 1990.
8. Y. Huang and T. S. Huang. Model-based human body tracking. In *International Conference on Pattern Recognition*, 2002.
9. S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated motion. In *International Conference on Automatic Face and Gesture Recognition*, pages 38–44, Killington, Vermont, 1996.
10. S. Khan and M. Shah. Tracking people in presence of occlusion. In *Asian Conference on Computer Vision*, Taipei, Taiwan, 2000.
11. L. Lasdon and A. Waren. *GRG2 User's Guide*, 1989.
12. D. Morris and J. Rehg. Singularity analysis for articulated object tracking. In *Computer Vision and Pattern Recognition*, 1998.
13. S. Park and J.K. Aggarwal. Segmentation and tracking of interacting human body parts under occlusion and shadowing. In *IEEE Workshop on Motion and Video Computing*, pages 105–111, Orlando, FL, 2002.
14. W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes*. Cambridge University Press, Cambridge, England, 1986.
15. R. Rosales and S. Sclaroff. Inferring body pose without tracking body parts. In *Computer Vision and Pattern Recognition*, pages 721–727, Hilton Head Island, South Carolina, 2000.
16. L. Salgado, N. Garcia, J. Menedez, and E. Rendon. Efficient image segmentation for region-based motion estimation and compensation. *IEEE Trans. Circuits and Systems for Video Technology*, 10(7):1029–1039, 2000.
17. H. Sidenbladh, M. Black, and D. Fleet. Stochastic tracking of 3d human figures using 2d image motion. In *ECCV (2)*, pages 702–718, 2000.