Multicamera Tracking of Articulated Human Motion Using Shape and Motion Cues
Aravind Sundaresan and Rama Chellappa, Fellow, IEEE

Abstract—We present a completely automatic algorithm for initializing and tracking the articulated motion of humans using image sequences obtained from multiple cameras. A detailed articulated human body model composed of sixteen rigid segments that allows both translation and rotation at joints is used. Voxel data of the subject obtained from the images is segmented into the different articulated chains using Laplacian Eigenmaps. The segmented chains are registered in a subset of the frames using a single-frame registration technique and subsequently used to initialize the pose in the sequence. A temporal registration method is proposed to identify the partially segmented or unregistered articulated chains in the remaining frames in the sequence. The proposed tracker uses motion cues such as pixel displacement as well as 2-D and 3-D shape cues such as silhouettes, motion residue, and skeleton curves. The tracking algorithm consists of a predictor that uses motion cues and a corrector that uses shape cues. The use of complementary cues in the tracking alleviates the twin problems of drift and convergence to local minima. The use of multiple cameras also allows us to deal with the problems due to self-occlusion and kinematic singularity. We present tracking results on sequences with different kinds of motion to illustrate the effectiveness of our approach. The pose of the subject is correctly tracked for the duration of the sequence as can be verified by inspection.

I. INTRODUCTION

HUMAN pose estimation and tracking from video sequences, or motion capture, has important applications in a variety of fields such as biomechanical and clinical analysis, human computer interaction, and animation. Current techniques use marker-based techniques, which involve the placement of markers on the body of the subject and capturing the movement of the subject using a set of specialized cameras. The use of markerless techniques eliminates the need for the specialized equipment as well as the expertise and time required to place the markers. It can also potentially measure the pose using anatomically appropriate models rather than estimating them from a set of markers. Different applications have different needs and use single or multiple cameras to estimate human pose and this problem has received much attention in the image processing and computer vision literature in both the monocular [1]–[4] and multiple camera cases [5]–[9]. A survey of a number of important pose estimation methods developed in the past decade may be found in [10]–[12].

The typical steps in motion capture [13] are (1) model estimation, (2) pose initialization, and (3) tracking. Model estimation is the process of estimating the parameters of the human body model such as the shape of the body segments and their articulated structure. Pose initialization refers to the estimation of the pose given a single frame.1 Pose tracking refers to the estimation of a pose in the next frame, given the pose in the current frame. Both (2) and (3) perform pose estimation, but the methods employed are usually different and we list them separately.

The articulated structure of the human body which is composed of a number of segments, each with its associated shape and pose, makes human pose estimation a challenging task. The complexity of the human body and the range of poses it can assume necessitate the use of a detailed model in order to represent its pose and to guide pose estimation. Body models typically incorporate both the shape of individual body parts and structural aspects such as the articulated connectivity and joint locations of the human body. Besides the sheer complexity of the human body, a common problem faced in image-based methods is that some parts of the body often occlude other parts (self-occlusion). It is, therefore, difficult to perceive and estimate motion in the direction perpendicular to the image plane when using images from a single camera. Morris and Rehg [14] refer to this problem as “kinematic singularity” and study it in some detail [14]. Monocular techniques suffer from the above problems of self-occlusion and “kinematic singularities” and multiple cameras are required to estimate pose in a robust and accurate manner.

A. Related Work

Gavrila and Davis [10], Aggarwal and Cai [15], and Moeslund and Granum [11] provide surveys of human motion tracking and analysis methods. Sigal and Black [12] provide a recent survey on human pose estimation. We list some of the important monocular and multicamera techniques in Sections I-A1 and I-A2 followed by a brief discussion of their limitations.

1) Monocular Methods: As mentioned earlier, monocular techniques suffer from a range of problems that mark them as unsuitable for markerless motion capture. Monocular methods can be classified according to the image cues used ranging from edges [2] and silhouettes [16] to 2-D image motion [4], [3]. In [17], the pose vector is estimated using support vector machines and kinematic constraints. The issue of model acquisition and

1By frame, we refer to image(s) obtained at a single time instant; it is one image in the monocular case and a set of images in the multicamera case.
initialization using images from a single camera is addressed in [18]. The problems of self-occlusion and kinematic ambiguities in monocular video have been addressed with limited success in [19]–[21]. Many of the monocular techniques predate multicamera methods and some have been extended to multiple cameras.

2) Multicamera Methods: Multicamera methods can also be broadly classified as shape-based and motion-based. Shape-based methods use 2-D shape cues such as silhouettes or edges [22]–[24] or 3-D shape cues such as voxels [5]–[8]. The voxel representation of a person provides cues about the 3-D shape of the person and is often used in pose estimation algorithms. Motion-based methods [25]–[27] typically use optical flow in the images to perform tracking. The motion-based methods estimate the change in pose and typically assume that the initial pose is available. On the other hand, shape-based methods use absolute cues and can be used to both initialize the pose given a single frame [5]–[8], or perform tracking [23], [24], [28], [29].

We first list automatic or semi-automatic methods to estimate and track the pose followed by some important multicamera techniques.

Mikić et al. [6] and Mündermann et al. [7] perform all the steps in the motion capture using voxel based techniques. They are, however, limited by the shortcomings of shape-based methods and in the case of [7], the model is not obtained automatically. Chu et al. [5] use volume data to acquire and track a human body model and Cheung et al. [9] use shapes from silhouette to estimate human body kinematics. However, in [5], no tracking is performed, while in [9], the subject is required to articulate one joint at a time in order toinitialize the pose.

The following techniques assume that an initial pose estimate is available and perform tracking using shape and motion cues. Yamamoto et al. [26] track human motion using multiple cameras and optical flow. Bregler and Malik [27] also use optical flow and an orthographic camera model. Gavrila and Davis [30] discuss a multiview approach for 3-D model-based tracking of humans in action. They use a generate-and-test algorithm in which they search for poses in a parameter space and match them using a variant of Chamfer matching. Kakadiaris and Metaxas [22] use silhouettes from multiple cameras to estimate 3-D motion. Delamarre and Faugeras [23] use 3-D articulated models for tracking with silhouettes. They use silhouette contours and apply forces to the contours obtained from the projection of the 3-D model so that they move towards the silhouette contours obtained from multiple images. Moeslund and Granum [24] perform model-based human motion capture using cues such as depth (obtained from a stereo rig) and the extracted silhouette, while the kinematic constraints are applied in order to restrict the parameter space in terms of impossible poses. Sigal et al. [28], [29] use nonparametric belief propagation to track in a multicamera set up.

Motion-based trackers suffer from the problem of drift; i.e., they estimate the change in pose from frame to frame and as a result the error accumulates over time. On the other hand, shape-based methods rely on absolute cues and do not face the drift problem but it is not possible to extract reliable shape cues in every frame. They typically attempt to minimize an objective function (which measures the error in the pose) and are prone to converge to incorrect local minima. Specifically, background subtraction or voxel reconstruction errors in voxel-based methods result in cases where body segments are missing or adjacent body segments are merged into one. We note that shape cues and motion cues are complementary in nature and it would be beneficial to combine these cues to track pose. We briefly describe our algorithm and discuss how it addresses the above limitations in the following section.

B. Algorithm Summary

We present a detailed articulated model and algorithms for estimating the human body model and initializing and tracking the pose in a completely automatic manner. The work presented in this paper is the second part of a larger body of work, namely, the complete automatic motion capture system. In order to place this work within the context of the larger body of work we first describe the contribution of our earlier paper titled “Model driven segmentation and registration of articulating humans in Laplacian eigenspace” [8] and then explain the contribution in this paper. The overview of our complete motion capture system is illustrated in Fig. 1.

In [8], we present an algorithm for segmenting volumetric representations (voxels) of the human body by mapping them to Laplacian Eigenspace. We also describe an application of this algorithm to human body model and pose estimation and provide experimental validation using both synthetic and real voxel data. Some of the key results of the above algorithm are illustrated in Fig. 2. Given a sequence of 3-D voxel data of human motion [Fig. 2(a)], the human body model and pose [Fig. 2(b)–(c)] are estimated using a sub-set of the frames in the
sequence. The human body model consists of rigid segments connected in an articulated tree structure. There are six articulated chains (the trunk, head and four limbs). The pose of each rigid segment is represented in general using a 6-vector (3 degrees of freedom for translation and 3 for rotation). However, in our work we constrain most of the joints to possess only rotational motion (3 degrees of freedom) and in general the complexity of the pose itself. A tracking error in the processing of the shape cues (e.g., voxel reconstruction) and in the tracking step with model estimation and pose initialization to obtain reasonable voxel reconstruction for the purpose of pose estimation as was also noted by Mün-dermann et al. [34]. We note that the prediction module of our tracker requires that the motion between frames be small enough so that pixel displacement can be estimated and the iterative pose estimation algorithm converges. We observe in our experiments that a frame rate of 30 fps suffices for normal human walking motion. The results of the tracking algorithm in sequences with different motions such as swinging arms in wide arcs, walking in a straight line and walking in circles are presented. The algorithm proposed in this paper can be used in a number of biomechanical applications, such as gait analysis as well as general human motion analysis.

The organization of the paper is as follows. We describe our human body model, corresponding pose vector and its estimation in Section II. The details of the algorithm are presented in Section III. We describe pose initialization, temporal spline registration and then describe the two-step tracking process. We also describe the smoothing step. Finally, we present results of our algorithm on three different sequences using real images captured from eight cameras in Section IV.

II. HUMAN BODY MODEL, POSE, AND TRACKING

We briefly describe our human body model in Section II-A and reconstruction of the subject using the body model parameters and the pose vector in Section II-B. We also describe some of the key modules of our tracking algorithm. The linear relation between the pixel velocity and pose velocity is derived and
the estimation of the change in pose from pixel displacements is described in Section II-C. The manner in which we use 3-D skeleton curves for tracking is described in Section II-D.

A. Human Body Model

We model the human body as consisting of six articulated chains, namely the trunk (lower trunk, upper trunk), head (neck, head), two arms (upper arm, forearm, palm) and two legs (thigh, leg, foot) as illustrated in Fig. 2(c). Our model is based on the underlying skeletal structure and flexibility of the human body. Each rigid segment is represented by a tapered super-quadric. The model consists of the joint locations and parameters of the tapered super-quadrics describing each rigid segment. The model can be simplified to a skeleton model using just the axis of the super-quadric as illustrated in Fig. 2(b). The recovery of the human body model is described in detail in [8], [32].

B. Description of Pose Vector

Let \( G_{ij} \in \mathbb{R}^{4 \times 4} \) be a transformation matrix in homogeneous 3-D coordinates consisting of a rotational component, \( \omega_{3 \times 1} \), and a translational component, \( \mathbf{p}_{3 \times 1} \). The pose vector for a single body segment consists of both components and is given by \( \mathbf{p} = (\mathbf{p}^T \omega^T)^T \). \( G \) is expanded as

\[
G(\mathbf{p}, \omega) = \hat{\varphi} = \begin{bmatrix} R & \mathbf{p} \\ 0 & 1 \end{bmatrix}, \quad \text{where} \quad R = e^{\begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix}}.
\]

(1)

The \( \wedge \) (hat) operator is described in [35] and maps a 6 \( \times \) 1 pose vector to the corresponding 4 \( \times \) 4 coordinate transform matrix. The \( \vee \) (vee) operator is the inverse of the \( \wedge \) operator and maps a 4 \( \times \) 4 coordinate transform to a 6 \( \times \) 1 pose vector, i.e., \( (\hat{\varphi})^\vee = \varphi \). We hereafter drop the dependency on pose vector for brevity. The articulated nature of the body is illustrated in Fig. 3. The lower trunk is the root of the kinematic chain and all body segments are attached to the root in a kinematic chain. Each body segment has six degrees of freedom (DoF) in general and its pose relative to its parent is described using the above pose vector.

We first define a world coordinate frame that is fixed for the entire experiment. The full body pose is computed with respect to this coordinate frame. The pose of each body segment is described by a combination of body model and pose parameters. We use the superscripts \( S \) and \( P \) to denote model structure and pose parameters respectively. For instance, \( \mathbf{p}^S \) is a joint location and is part of the body model, while \( \mathbf{p}^P \) is the translational pose at the joint and is part of the pose vector. Consider two segments, \( i \rightarrow i-1 \) and \( j \rightarrow i \) in Fig. 3, where segment \( i \rightarrow i-1 \) is the parent of segment \( i \). Segment \( i \) is connected to its parent at joint \( i \), whose location is given by \( \mathbf{p}^S(i) \) in the coordinate frame of the parent. We hereafter use the word “frame” as an abbreviation for “coordinate frame.” The pose of segment \( i \) with respect to its parent (segment \( i \rightarrow i-1 \)) is \( \varphi(i) = (\mathbf{p}(i) \omega(i))^T \). The complete transformation between segment \( i \) and segment \( i \rightarrow i-1 \) is, therefore, given by

\[
G_{(i-1)i} = G_{(i-1)i} = \begin{bmatrix} \mathbf{p}^S(i) \\ 0 \end{bmatrix} G_{(i-1)i} \varphi(i).
\]

(2)

\( G_{ij} \) represents a transformation matrix of a point from the coordinate frame of segment \( j \) to the coordinate frame of segment \( i \). \( G_{01} \) represents the transformation matrix of the root of the kinematic chain (index 1) with respect to the world coordinate frame (index 0). The pose of the \( i \)th segment in Fig. 3 with respect to the world coordinate frame is, therefore, given by

\[
G_{0i} = G_{01} G_{12} \cdots G_{(i-1)i}.
\]

(3)

For a strictly articulated body, the translation component of the pose at all joints is zero, i.e., \( \mathbf{p}(i)^P = 0 \) \( \forall i > 1 \). However, we allow limited translation at certain joint locations such as the shoulder (a “compound” joint that cannot be represented by a single rotational joint) to better model its complexity. We set \( \|\mathbf{p}(i)^P\| < P_{\text{MAX}} \) where \( \mathbf{p}(i)^P \) denotes special joints. Our human body model consists of sixteen rigid segments. The pose of segment \( i \) is given by \( \varphi(i) = (\mathbf{p}(i)^P \omega(i))^T \) and the full body pose is given by \( \Phi = (\varphi(1)^T \cdots \varphi(16)^T)^T \).

C. Tracking Pose Using Pixel Displacement

In this section, we describe how we estimate the change in pose of an articulated body using pixel displacement. To begin with, we introduce the 6 \( \times \) 1 pose velocity vector \( \dot{\mathbf{e}} \). We can describe any six DoF transformation matrix at time \( t \) as \( e^T \dot{\mathbf{e}} \) [35]. The pose velocity is so called as it can be considered as the instantaneous velocity of the pose vector. The instantaneous pose velocity at \( t = 0 \) is given by

\[
\frac{d}{dt}(e^T \dot{\mathbf{e}})|_{t=0} = \dot{\mathbf{e}} e^T \dot{\mathbf{e}}|_{t=0} = \dot{\mathbf{e}}.
\]

(4)

We derive the relation between the pose velocity vector and the 3-D pixel velocity as well as the 2-D pixel velocity. Finally we describe the algorithm to estimate the pose velocity from the pixel displacement. We denote the transformation of frame \( j \) (attached to segment \( j \)) with respect to frame \( i \) (attached to segment \( i \)) at time \( t \) by \( g_{ij}(t) \). We can then express \( g_{ij}(t) \) as

\[
g_{ij}(t) = g_{ij}(0) \dot{\mathbf{e}} = G_{ij} \dot{\mathbf{e}}
\]

(5)
where $G_{ij} = g_{ij}(0)$ is defined earlier. We use $g$ to denote time varying matrices and $G$ to denote constant matrices. We note that $\hat{q}^i$ can be expanded using the Taylor series as

$$e^{\hat{q}^i} = I + \hat{q}^i + \frac{1}{2!} (\hat{q}^i)^2 + \frac{1}{3!} (\hat{q}^i)^3 + \cdots. \quad (6)$$

Let us consider a point $q$ attached to segment $j$. Its coordinates in frame $i$ and frame $j$ are given by $q^{(i)}$ and $q^{(j)}$, respectively. We then have

$$q^{(j)}(t) = g_{ij}(t)q^{(i)}(t) = G_{ij}\hat{e}^i q^{(j)}(t). \quad (7)$$

We consider motion at $t = 0$ without loss of generality. Considering the instantaneous velocity $\dot{q}$ of the point $q$ in frame $i$, we have

$$\dot{q}^{(i)} = \frac{d}{dt} q^{(i)} = g_{ij}\dot{q}^{(j)} + g_{ij}\dot{q}^{(j)} = g_{ij}\dot{q}^{(j)} \quad (8)$$

where the second equation follows because the point is attached to frame $j$ and, therefore, $\dot{q}^{(j)} = 0$. Substituting (5) in (8), we get (9), (10) follows where $\Upsilon(q)$ is defined in (11)

$$\dot{q}^{(j)}(t) = G_{ij}\hat{e}^i q^{(j)} \quad (9)$$

$$\dot{\Upsilon}(q) = \begin{pmatrix} 1 & 0 & 0 & 0 & -q_3 & q_2 \\ 0 & 1 & 0 & q_3 & 0 & -q_1 \\ 0 & 0 & 1 & -q_2 & q_1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (10)$$

Assuming there are a total of $m$ segments, and given a point $q$ on the $i$th segment, we have

$$q^{(0)} = g_{0i}q^{(i)} = g_{0i}g_{12}\cdots g_{(i-1)i}q^{(i)}. \quad (12)$$

It follows that

$$\dot{q}^{(0)} = (g_{0i}g_{12}\cdots g_{(i-1)i} + g_{0i}g_{12}\cdots g_{(i-1)i} + \cdots + g_{0i}g_{12}\cdots g_{(i-1)i}) q^{(i)} \quad (13)$$

$$= g_{0i}g^{(i)} + g_{0i}g^{(2)}\cdots g_{(i-1)i} + \cdots + g_{0i}g^{(i)} \quad (14)$$

$$= g_{0i}\Upsilon^{(i)} q^{(i)} + g_{0i}\Upsilon^{(2)} q^{(i)} + \cdots + g_{0i}\Upsilon^{(i)} q^{(i)} \quad (15)$$

where $\Upsilon = (\epsilon^{(1)})^{v} \cdots (\epsilon^{(m)})^{v}$ and $F(\Phi, q)$ follows from (17).

Let the pose at time $t = 0$ and $t = 1$ be $\Phi(0)$ and $\Phi(1)$ be $\Phi(0)$ and $\Phi(1)$, respectively. The pose at $t = 1$ for each segment $i$ in the body is then given by

$$\Phi(1) = \Phi(0)^{1} \cdots \Phi(0)^{m} \quad (19)$$

We can represent the set of operations (19) using the abbreviated notation

$$\Phi_{t+1} = \Phi_2 \Xi \quad (20)$$

where the upper-case Greek letters $(\Phi, \Xi)$ refer to the vector stack of the poses of the individual segments represented by lower-case Greek letters $(\varphi, \xi)$.

We have shown in [36] that if we use a perspective projection to project a 3-D point onto the camera, the resulting pixel velocity is still a linear function of the pose velocity. Let $P_{3 \times 4} = P'_{3 \times 2}$ be the projection matrix, then the pixel location in homogeneous coordinates is given by $q^{(c)} = Pq^{(0)}$ and the actual pixel coordinates are given by $u$ as

$$u = \begin{pmatrix} u_2 \\ u_2 \end{pmatrix} = \frac{1}{q_3^{(c)}} \begin{pmatrix} q_1^{(c)} \\ q_2^{(c)} \end{pmatrix} = \frac{1}{P'q^{(0)}} \begin{pmatrix} P'_{1} \end{pmatrix} q^{(0)}. \quad (21)$$

We obtain the pixel velocity by differentiating (21) as

$$\dot{u} = \frac{1}{P'q^{(0)}} \left( \begin{pmatrix} P'_{1} \end{pmatrix} q^{(0)} - \frac{1}{P'_{3}q^{(0)}} \left( \begin{pmatrix} P'_{1} \end{pmatrix} q^{(0)} \right) \right) \quad (22)$$

$$= E(P, q)\dot{q}^{(0)} \quad (23)$$

$$= E(P, q)F(\Phi, q)\Xi \quad (24)$$

We represent the matrix in (22) as $E(P, q)$ in (23). We thus combine (18) and (23) to express the 2-D pixel velocity as a linear function of the 3-D pose velocity, $\Xi$ in (24). We can estimate $\Xi$ from the pixel velocity by inverting (24), then we can use (20) to compute the new pose $\Phi_{t+1}$ from $\Phi_t$. However, we can only measure pixel displacement from the images, and, hence, we use a first-order approximation of the pixel velocity.

Given a set of points, we compute the projection of each of these points for all the cameras using the pose $\Phi$. We call this stacked vector $C(\Phi)$. We also compute the pixel displacement matrix $D(\Phi) = E(P, q)F(\Phi, q)\Xi$ and $C(\Phi)$ are functions of both the 3-D point coordinates and the projection matrices besides $\Phi$, but as these are fixed for a given frame, we do not explicitly denote them for the sake of simplicity. For a set of $n$ points, we, therefore, have

$$\begin{pmatrix} u_1 \\ \vdots \\ u_n \end{pmatrix} = C(\Phi) \begin{pmatrix} \dot{u}_1 \\ \vdots \\ \dot{u}_n \end{pmatrix} = \Phi_{t+1} \Xi \quad (25)$$

The state vector in our state-space formulation is $\Phi_t$ and the state update and observation equations are given by (26) and (27), where $w_{t+1}$ is the measurement noise.
\[
\dot{\Phi}_{t+1} = \Phi_t \Xi_t \\
\Delta u_t = u_{t+1} - u_t + w_{t+1} \approx D(\Phi) \Xi + w_{t+1}.
\]

We note that our system is similar to an iterated extended Kalman filter but the plant noise in our case (\(\Xi_t\)) is multiplicative and it is not straight-forward to extend the IEKF in our system. Equation (27) follows from the first order Taylor series approximation

\[
u_{t+1} = u_t + \dot{u}_t + \frac{1}{2} \ddot{u}_t + \cdots \approx u_t + \dot{u}_t.
\]

We then use Algorithm 1 to estimate \(\Phi_{t+1}\) given \(\Phi_t\) and the pixel displacement between \(t\) and \(t+1\). We have several pixel displacement measurements from multiple cameras and the estimation equation is highly over-constrained, and, therefore, we can obtain a least squares estimate. We also find that the multiview constraints are able to overcome kinematic singularities and occlusions which are the bane of monocular pose estimation. We observe that we compute the change of pose (\(\Xi_t\)) between two frames. The constant \(\Xi\) assumption is perfectly valid when we consider only two frames. The approximation we make is that we estimate the pose velocity which is a first order approximation of the pixel velocity which should be used. We use an iterative estimation algorithm to compensate for the approximation but we note that a higher frame rate is required for faster motions than those presented in the experiments.

**Algorithm 1 Compute 3-D Pose From Pixel Displacement**

**Require:** Pose at time \(t\), \(\Phi_t\) and pixel displacement between \(t\) and \(t+1\), \(\Delta u_t\)

1: set \(\Phi^{(0)} = \Phi_t\)
2: for \(k = 0 : \) maximum iterations - 1 do
3: let \(\Delta u^{(k)} = \Delta u_t - (C^{(k)}(\Phi^{(k)}) - C^{(k)}(\Phi_t))\)
4: compute \(\Xi^{(k)} = (D^{(k)}(\Phi^{(k)}) D^{(k)}(\Phi^{(k)}) \Delta u^{(k)})^{\top}\)
5: update \(\Phi^{(k+1)} = \Phi(k)^{\top} \Xi^{(k)}\)
6: end for
7: set \(\Phi_{t+1} = \Phi^{(k+1)}\)

**D. Tracking Pose Using Skeleton Curves**

In this section, we describe a key module in the tracking of the pose using 3-D shape cues (skeleton curves). As described earlier and illustrated in Fig. 4, we can segment voxel data [Fig. 4(a)] into different articulated chains and register them to the human body model [Fig. 4(b)]. We can obtain a skeleton curve for each segmented articulated chain represented by uniformly-spaced points on the curve [Fig. 4(c)]. The skeleton model corresponding to the estimated pose is illustrated in Fig. 4(d). In order to determine the pose that best fits the skeleton curve, we first define a distance measure between the skeleton curve [Fig. 4(c)] and the skeleton model [Fig. 4(d)] computed from the pose. We assume that an initial estimate of the pose is available so that we can iteratively refine the estimate to minimize this distance. We note that the skeleton curve for a frame consists of six curves registered to the six articulated chains of the human body model.

We compute the distance by considering each chain independently as described in the following paragraph.

Consider a set of \(n\) ordered points \(x_1, x_2, \ldots, x_n\), on a skeleton curve corresponding to the arm (see Fig. 5). The skeleton model for the arm consists of three line segments: \(L_1, L_2,\) and \(L_3\). We compute the distance, \(e_i^j\), between \(x_i\) and the closest point on line segment \(L_j\) and assign each point to a line segment. Since the set of points on the skeleton curve is ordered, we impose the constraint that the above assignment is performed in a monotonic manner, i.e., points \(x_1, \ldots, x_{n_1}\) are assigned to \(L_1\), points \(x_{n_1+1}, \ldots, x_{n_2}\) are assigned to \(L_2\) and points \(x_{n_2+1}, \ldots, x_n\) are assigned to \(L_3\). For a given value of \(n_1, n_2\) is chosen so that the distance between points \(x_{n_1}\) and \(x_{n_2}\) is equal to the length of line segment \(L_2\). For the above assignment, the distance between the skeleton curve is given by the vector \((e_1^1 \cdots e_{n_1}^1 e_2^2 \cdots e_{n_2}^2 e_3^3 \cdots e_n^3)^\top\). The 3-D pose of the articulated chain as well as indices \(n_1\) and \(n_2\) are chosen so as to minimize the sum of the elements in the vector which is given by

\[
\sum_{i=1}^{n_1} e_i^1 + \sum_{i=n_1+1}^{n_2} e_i^2 + \sum_{i=n_2+1}^{n} e_i^3.
\]

**III. ALGORITHM**

We present in this section the details of our pose tracking algorithm including the preprocessing and pose initialization steps. The preprocessing includes using images obtained from multiple cameras to compute silhouettes and voxels and is described in Section III-A. The parameters of the human body model are computed as described in [32]. We assume that we are
able to perform single-frame registration in at least one frame in the sequence and initialize the pose using the method described in [31]. Typically, the pose can be initialized in some of the frames in the sequence but the single-frame registration is unsuccessful in the majority of the frames and we are left with unregistered skeleton curves. We propose a temporal registration scheme by which means we register skeleton curves by exploiting their temporal relation. Methods for pose initialization and temporal skeleton curve registration are described in Section III-B. We then describe our tracking algorithm that tracks the pose in two steps; the prediction step using motion cues, and the correction step using 2-D and 3-D shape cues in Section III-C. We also describe an optional smoothing step in Section III-D.

A. Preprocessing

We use images obtained from \( N = 8 \) calibrated cameras. We perform simple background subtraction to obtain foreground silhouettes as shown in Fig. 6. In order to compute the voxels, we project points on a 3-D grid (in the volume of interest) to all the camera images. All points that are projected to image coordinates that lie inside the silhouette in all the images are considered to be part of the subject. In general, we could consider points that lie inside the silhouette in at least \( N - M \) images, where \( M \) could take values \( 0, 1, 2, \ldots \) depending on the number of cameras in use. A nonzero value of \( M \) lends robustness to background subtraction errors if there are a large number of cameras \((N > 10)\). We set \( M = 0 \) in our experiments. The voxel reconstruction results using the silhouettes in Fig. 6(b) are presented in Fig. 6(c).

B. Pose Initialization and Temporal Registration

We perform segmentation of the voxel data using Laplacian Eigenmaps to obtain the different articulated chains [8], [31]. The method maps voxels on each articulated chain to a smooth 1-D curve in Laplacian Eigenspace. We can then segment voxels as belonging to different curves (or articulated chains) and also register them. For each segmented articulated chain we compute the skeleton curve using smoothing splines as described in [31]. The method to initialize the pose of the subject using the registered skeleton curve is presented in Section III-B1. An example of a successfully segmented and registered frame is presented in Fig. 7. However, the single frame registration method does not succeed in all frames due to errors in voxel reconstruction or segmentation, examples of which are presented in Figs. 8 and 9.

We present a temporal registration algorithm to register skeleton curves in such frames in Section III-B2.

1) Pose Initialization: The pose is initialized for a completely registered frame as follows. The skeleton curve is sampled at regular intervals to obtain a set of ordered points for each body chain (trunk, head, two arms and two legs). The sampled skeleton curve is illustrated in Fig. 7(c). We choose an intersample distance of 20 mm as a trade-off between the computational cost of denser sampling and the poor spatial resolution of sparser sampling.

The pose is computed using the skeleton curves and is initialized in two steps. First, the pose of the trunk is determined and second, the pose of the remaining five articulated segments is computed. The \( z \)-axis of the trunk is aligned with the skeleton curve of the trunk as marked in Fig. 7(d). The \( y \)-axis of the trunk is parallel to the right-left vector which is set to be the average of the vectors from the right to left shoulder joint and from the right to left pelvic joint on the skeleton curve marked in Fig. 7(d). The \( x \)-axis points in the forward direction which is determined using the direction of the feet and is orthogonal to the computed \( yz \) plane. The \( yz \) axis orientation that describes the pose of the trunk is illustrated in Fig. 7(e). Once the trunk pose has been estimated, the joint locations at the hips, shoulders and neck are fixed. It is then possible to estimate the pose of each of the articulated chains independently. The objective is to compute the pose of the skeleton model so that the distance between the points on the skeleton curve and the skeleton model is minimized as described in Section II-D. The initial estimate of the pose is illustrated in Fig. 7(f).

2) Temporal Skeleton Curve Registration: Two examples where registration of skeleton curves to articulated chains in a single frame fails are illustrated in Figs. 8 and 9. In one of the examples, the head is missing due to errors in background subtraction. In the other seven skeleton curves are discovered instead of six. We, therefore, introduce a temporal registration scheme which exploits the proximity of the skeleton curves belonging to the same body segment in temporally adjacent frames. Given two frames at \( t_0 \) and \( t_1 > t_0 \) where single-frame registration is successful, we perform temporal registration in all the frames between \( t_0 \) and \( t_1 \). Let \( S_A = \{ x_1^A, x_2^A, \ldots , x_{n_A}^A \} \) and \( S_B = \{ x_1^B, x_2^B, \ldots , x_{n_B}^B \} \) be the set of points belonging to skeleton curves \( S_A \) and \( S_B \) respectively. The distance between skeleton curves \( S_A \) and \( S_B \) is given by

\[
d(S_A, S_B) = \frac{1}{n_A + n_B} \left( \sum_{i=1}^{n_A} \min_j \left( \| x_i^A - x_j^B \| \right) + \sum_{i=1}^{n_B} \min_j \left( \| x_i^B - x_j^A \| \right) \right).
\]

Let \( S_i^A \) and \( R_i^A \) represent the unregistered and registered skeleton curves for the \( i \)-th articulated chain at time instant \( t \). The temporal skeleton curve registration algorithm is listed in Algorithm 2. We typically set \( d_{\text{threshold}} = 50 \) mm. This threshold is based on the maximum distance a chain can move between two frames and the intra curve distance within a given frame.

**Algorithm 2 Temporal Skeleton Curve Registration**

![Image](image-url)
We use Algorithm 2 to perform reverse temporal registration as well, i.e., we start at $t = t_1$ and proceed backwards in time to $t = t_0$. The reverse registration is necessary because if there is a gap in the forward temporal registration for any of the six articulated chains, then that chain is unlikely to be registered from that point onwards. Any skeleton curve that is not registered to the same articulated chain in the forward and reverse temporal registration process is not used in the tracking.

C. Pose Tracking

Our tracking algorithm consists of a prediction step and a correction step that are described in Sections III-C1 and III-C2, respectively. The overview of the tracking algorithm is presented in Algorithm 3.

Algorithm 3 Tracking Algorithm

```plaintext
1: for time $t = t_0 : t_1$ do
2: /* predict pose at time $t + 1 */
3: compute 2-D pixel displacement between frames $t$ and $t + 1$
4: estimate 3-D pose using pixel displacement
5: /* correct pose at time $t + 1 */
6: for chain $i = 1 : 6$ do
7: if 3-D shape cues are available for chain $i$ then
8: correct pose using 3-D shape cues (skeleton curves).
9: else
10: correct pose using 2-D shape cues (silhouettes and motion residues).
11: end if
12: end for
13: end for
```

1) Prediction Using Motion Cues: In order to estimate the motion of each of the body segments, we first project the body segment onto each image. We call this step pixel-body registration. We then compute the pixel displacement for each body segment in each image using the motion model for a rigid segment. The pixel displacement for a set of bodies in all the images is then stacked in a single matrix equation which we use to estimate the change in 3-D pose.
is the displacement. 

means there is no motion. 

(the vector of 
is the observed image at time 
is the rotation and 
is the 2-D location of the joint of the body segment.

be 

for 
denotes the imposed

warped according to the estimated

for estimated

. (d) The combined error image. (e) Error image with the mask corresponding

time

Fig. 13. Obtaining unified error image for the forearm. (a) The silhouette at

time \( t + 1 \). (b) A magnified view of the silhouette. (c) The motion residue at time 

t. (d) The combined error image. (e) Error image with the mask corresponding to 

the segment whose pose we are trying to correct.

segments are projected onto the same pixel the registration ambiguity is resolved by using the depth. We thus register each pixel to the body segment it belongs to and determine its 3-D coordinates. Fig. 10 illustrates the projection of the body segments onto images from two cameras. Different colors denote different body segments.

b) Estimating Pixel Displacement: We use a parametric model for the motion of pixels belonging to the same segment in each image. The displacement at pixel \( \mathbf{u} \) is a function of \( \mathbf{\psi} = [\Delta, \theta, s] \) where \( \Delta \) is the displacement, \( \theta \) is the rotation and \( s \) is the scale parameter and is given by

\[
\delta(\mathbf{u}, \mathbf{\psi}) = s \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} \begin{pmatrix} u - u_0 \end{pmatrix} + \Delta \tag{31}
\]

where \( u_0 \) is the 2-D location of the joint of the body segment. We find that the above parametric representation is more robust than an affine motion model and we can also set meaningful upper and lower bounds on each parameter. Let \( \mathbf{u}_1, \ldots, \mathbf{u}_n \) be the pixels registered to a given segment. We compute the value of \( \mathbf{\psi} \in [\psi_0 - \psi_{\delta \psi}, \psi_0 + \psi_{\delta \psi}] \) for each segment that minimizes the residue given by \( \delta \mathbf{e} \mathbf{e} \). The value \( \psi_{\delta \psi} \) denotes the imposed bounds on the motion and the \( j \)th element of \( \mathbf{e} \) (the vector of pixel residues) is given as

\[
e_j = I_t(\mathbf{u}_j) - I_{t+1}(\mathbf{u}_j + \delta(\mathbf{u}_j, \mathbf{\psi})) \tag{32}
\]

where \( I_t \) is the observed image at time \( t \) in one of the cameras. A value of \( 0 \) for \( \mathbf{\psi} \) means there is no motion.

Fig. 11 illustrates the pixel displacement computations. Once we have estimated the motion parameters, we can estimate the corresponding “motion residue” which is defined as the difference between \( I_{t+1} \) and \( I_t \) warped according to the estimated motion. If the actual motion of a pixel agrees with the estimated

Fig. 10. Pixel registration: (a) view 1, (b) view 2.

Fig. 11. (a) Smoothed image with the foreground mask. (b) Motion residue for 

\( \mathbf{\psi} = 0 \). (c) The estimated pixel displacement for the mask. (d) Motion residue 

for estimated \( \mathbf{\psi} \) that results in the pixel displacement in (c).

Fig. 12. We estimate the motion from the base of the kinematic chain, i.e., the 

trunk and propagate the motion in further steps. The segments for which the 

pose is computed at a given stage is colored in black. (a) Step 1. (b) Step 2. (c) 

Step 3.

Fig. 13. Obtaining unified error image for the forearm. (a) The silhouette at 
time \( t + 1 \). (b) A magnified view of the silhouette. (c) The motion residue at time 
t. (d) The combined error image. (e) Error image with the mask corresponding to 
the segment whose pose we are trying to correct.

a) Pixel-Body Registration: In order to determine the correspondence between each pixel and the body segments we represent each body segment as a triangular mesh and project it onto each image. The depth at each pixel is determined by interpolating the depths of the triangle vertices. When multiple
motion, then the motion residue for that pixel is zero and otherwise it is some nonzero value. We note that the motion \( \mathbf{\Phi} = \mathbf{0} \) [Fig. 11(b)] agrees with the motion of the stationary background pixels. However, it does not agree with the motion of the foreground pixels. Fig. 11(c) denotes the estimated pixel displacement for the body segment under consideration. Fig. 11(d) is the motion residue for the estimated \( \mathbf{\Phi} \). We note that the estimated motion agrees with the actual motion for the foreground pixels (in the mask) but not for the background pixels, i.e., the motion residue for the pixels in the mask is almost zero. Thus, the motion residue provides us with a rough delineation of the location of the body segment, even when the original mask does not exactly match the body segment.

c) Pose Prediction: We predict the pose \( \mathbf{\Phi}_{t+1} \) at time \( t+1 \) given the pose \( \mathbf{\Phi}_t \) at time \( t \) and the pixel displacement computed above as described in Algorithm 1 in Section II-C. While our body model and our pose estimation algorithms allow rotation and translation at each joint, we set the translational component to zero at most joints as we find that in practice the estimation is more robust when the number of translational parameters is small [37]. We estimate the pose of the subject in multiple steps, starting at the root of the kinematic chain as illustrated in Fig. 12. In the first step, we estimate the pose for the segments belonging to the trunk, in the second we include the first segment in all the articulated chains, and in the final step we estimate the pose for all the segments except the trunk. The translational component of all segments is set to zero except for the base body and the shoulder joint. The base body is allowed to translate freely and the translation at the shoulder joint, \( \mathbf{p}_{\text{SHOULDER}} \) is constrained so that \( ||\mathbf{p}_{\text{SHOULDER}}|| \leq 20 \) mm.

2) Correction Using 3-D and 2-D Shape Cues: The pose can be corrected for all the articulated chains in a given frame that have been registered using 3-D shape cues (skeleton curves). The pose parameter search space is centered and bounded around the pose predicted using motion cues. In the absence of 3-D shape cues, we use 2-D shape cues in the form of silhouettes and motion residues. Thus, the algorithm adapts itself to use available spatial cues.

We have observed earlier that the motion residue for a given segment provides us with a region that helps us to spatially delineate the segment. We now combine it with the silhouette as illustrated in Fig. 13 to form an error image for that segment. The error image is the sum of the silhouette and motion residue and is computed for each camera and each segment along with a mask for the body segment as illustrated in Fig. 13(e). This error image can be used to compute an objective function in terms of the 3-D pose of the segment. Given any 3-D pose of the segment we can project the segment onto each image to obtain a mask for the segment [Fig. 13(e)]. The objective function is computed by summing the pixels of the error image that lie in the mask. Our estimate of the 3-D pose is the value that minimizes this objective function in all the images. The objective function is optimized in a pose parameter space that is centered around the predicted pose using the \textit{lsqnonlin} function in the Matlab nonlinear optimization toolbox. We illustrate the results of pose correction for the above example in Fig. 14. The red line represents the initial position of the axis of the body segment and the cyan line represents the final position. The final mask location is denoted in blue and we note that it is well aligned with the silhouette.

D. Smoothing

It is often beneficial to perform temporal smoothing on the pose vector as it typically improves the performance of the algorithm. We propose an optional smoothing step that acts on the pose of the root segment of the kinematic chain. It is difficult to smooth the entire pose vector due to the articulated constraints between the segments, and we therefore restrict the smoothing to the pose of the trunk segment (root) as it has an impact on the pose of all the body segments. We smooth the pose estimated from the skeleton curves using the smoothing spline function \texttt{csaps} in the Matlab Spline Toolbox. The trunk location is interpolated for frames missing the trunk skeleton curve. The translational components of the pose of the trunk for one of the test sequences is presented in Fig. 16. The translational components are given by

\[
\mathbf{p} = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix}.
\]

IV. EXPERIMENTAL RESULTS

We performed tracking on sequences where the subject performs different kinds of motion. The experiments were performed using gray-scale images obtained from eight cameras with a spatial resolution of 648 x 484 at a frame rate of 30 frames per second. The external and internal camera calibration parameters for all the cameras were obtained using the camera calibration algorithm of Svoboda [38] and a simple calibration device to compute the scale and the world coordinate frame. We present results for three sequences that include the subject walking in a straight line (65 frames, 2 s) in Fig. 17, swinging

![Fig. 16. Raw and smooth translational components represented by dots and lines, respectively. (a) \( p_1 \) (x component), (b) \( p_2 \) (y component), (c) \( p_3 \) (z component).](image-url)
the arms in wide arcs (300 frames, 10 s) in Fig. 18, and walking in a circular path (300 frames, 10 s) in Fig. 19. Fig. 15 illustrates the motion of the base body in the world coordinate frame in the three sequences. Our experiments show that using only motion cues for tracking causes the pose estimator to drift and lose track eventually, as we are estimating only the difference in the pose. This underlines the need for “correcting” the predicted pose using spatial cues and we observe that the “correction” step of the algorithm prevents drift in the tracking. We illustrate the results of the tracking algorithm by super-imposing the tracked body model onto the image for two of the eight cameras. The estimated pose of the subject is super-imposed on the images and the success of the tracking algorithm is determined by visual inspection. It is not possible to obtain an objective measure of the pose as the actual pose is not available. The full body pose is successfully tracked in the three sequences as can be observed in the super-imposed video sequences. Selected frames from different cameras are presented in Figs. 17–19.

V. CONCLUSION

We presented a complete pose initialization and tracking algorithm using a flexible and full human body model that allows translation at complex joints such as the shoulder joint. The human body model is automatically estimated from the sequence using the algorithm presented in [8], [32]. Pose initialization is performed based on single frame segmentation and registration [8], [31] of voxel data. An algorithm to perform temporal registration of partially segmented voxels for tracking was also suggested. We used both motion cues and shape cues such as skeleton curves obtained from bottom-up voxel segmentation as well as silhouettes and “motion residues” to perform the tracking. We presented results on sequences with different kinds of motion and observe that the several independent cues used in the tracker enable it to perform in a robust manner. The complete motion capture system has been written in Matlab and we note that currently the computational requirements of the
system are high primarily due to the number of cameras used in the processing and the inefficiency of the Matlab platform. The tracking process takes approximately 10–15 s per frame on a Pentium Xeon 2-GHz processor. Some of the most computationally intensive modules such as the projection of the human body model onto each of the images can be optimized and also made parallel and it is possible to optimize the system to operate at speeds of 1 frame per second or better. We anticipate that our motion capture system can be optimized and polished for use in a variety of important applications in biomechanical and clinical analysis, human computer interaction and animation.

REFERENCES


Aravind Sundaresan received the B.E. (Hons.) degree from the Birla Institute of Technology and Science, Pilani, India, in 2000, and the M.S. and Ph.D. degrees from the Department of Electrical and Computer Engineering, University of Maryland, College Park, in 2005 and 2007, respectively.

He is currently a computer scientist at the Artificial Intelligence Center at SRI International, Menlo Park, CA. His research interests are in pattern recognition, image processing, and computer vision, and, in particular, their application to markerless motion capture, real-time pose tracking in crowds, and 3-D scene modeling and navigation for mobile robots.
Rama Chellappa (S’78–M’79–SM’83–F’92) received the B.E. (Hons.) degree from the University of Madras, India, in 1975, and the M.E. (Distinction) degree from the Indian Institute of Science, Bangalore, in 1977. He received the M.S.E.E. and Ph.D. degrees in Electrical Engineering from Purdue University, West Lafayette, IN, in 1978 and 1981, respectively.

Since 1991, he has been a Professor of electrical engineering and an affiliate Professor of computer science at the University of Maryland, College Park. He is also affiliated with the Center for Automation Research (Director) and the Institute for Advanced Computer Studies (Permanent Member). In 2005, he was named a Minta Martin Professor of Engineering. Prior to joining the University of Maryland, he was an Assistant (1981-1986) and Associate Professor (1986-1991) and Director of the Signal and Image Processing Institute (1988-1990) at University of Southern California, Los Angeles. Over the last 28 years, he has published numerous book chapters, peer-reviewed journal, and conference papers. He has coauthored and edited many books in visual surveillance, biometrics, MRFs and image processing. His current research interests are face and gait analysis, 3-D modeling from video, image and video-based recognition, and exploitation and hyper spectral processing.

Prof. Chellappa has served as an Associate Editor of many IEEE Transactions, as a Co-Editor-in-Chief of Graphical Models and Image Processing, and as the Editor-in-Chief of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He served as a member of the IEEE Signal Processing Society Board of Governors and as its Vice President of Awards and Membership. He is serving a two-year term as the President of IEEE Biometrics Council. He has received several awards, including an NSF Presidential Young Investigator Award, four IBM Faculty Development Awards, an Excellence in Teaching Award from the School of Engineering at USC, two paper awards from the International Association of Pattern Recognition, and the Technical Achievement and Meritorious Service Awards from the IEEE Signal Processing Society and the IEEE Computer Society. At the University of Maryland, he was elected a Distinguished Faculty Research Fellow and a Distinguished Scholar-Teacher; he received the Outstanding Faculty Research Award from the College of Engineering and an Outstanding Innovator Award from the Office of Technology Commercialization. He is a Fellow the International Association for Pattern Recognition and the Optical Society of America. He has served as a General the Technical Program Chair for several IEEE international and national conferences and workshops. He is a Golden Core Member of the IEEE Computer Society and serving a two-year term as a Distinguished Lecturer of the IEEE Signal Processing Society.