Multi-camera Tracking of Articulated Human Motion using Shape and Motion Cues

Aravind Sundaresan, Student Member, IEEE, and Rama Chellappa, Fellow, IEEE

Abstract

We present a completely automatic framework and algorithm for initializing and tracking articulated motion of humans using video sequences obtained from multiple cameras. We use a detailed articulated human body model composed of sixteen rigid segments that allows both translation and rotation at joints. We use voxel data of the subject, which can be 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 remainder of the frames. We propose a tracking framework that uses motion cues, such as pixel displacement, as well as 2D and 3D shape cues (silhouettes, motion residue and skeleton curves) to perform tracking, given the initialized pose. The tracking framework 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. We present tracking results on different kinds of motion that illustrate the efficacy of the approach.

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 and has received much attention in the image processing and computer

This research was funded in part by NSF ITR 0325715.
The authors are with the Center for Automation Research, University of Maryland Institute for Advanced Computer Studies, and the Department of Electrical and Computer Engineering at the University of Maryland, College Park, MD 20742-3275, USA. Email: {aravinds,rama}@cfar.umd.edu.
vision literature. The articulated structure of the human body makes pose estimation and tracking a difficult task. It is necessary to use multiple cameras to deal with occlusion and kinematic singularities. It is also very helpful to use body models that incorporate both the shape of individual body parts and structural aspects such as the articulated connectivity and joint locations of the human body to guide the pose estimation. The typical steps in motion capture are model estimation, pose initialization and tracking. Much of the work in the past has focused on using either motion or shape cues for pose estimation. A number of algorithms assume that the pose is initialized and tackle the tracking problem. On the other hand, methods that estimate pose using a single frame, typically are not able to estimate the pose of the subject in all the frames in a sequence. One of the reasons is that initialization and tracking methods typically use different type of inputs. We present a pose estimation and tracking algorithm that combines both 3D and 2D shape cues as well as 2D image motion cues to initialize and track pose in a sequence using an articulated model of the subject. 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) as illustrated in Fig. 2. We represent the pose, \( \Phi \), in a parametric form as a vector of the poses of each segment. Each segment has, in general, six degrees of freedom (three translational and three rotational). We use a bottom up algorithm proposed in [1] to segment the voxel structure into different articulated chains. A top-down approach is then be used to register the segmented articulated chains to the different body segments in a single frame using a probabilistic approach. We select those frames where all six articulated chains have been segmented and the probability of registration is high. We can estimate the human body model and pose for these frames using the method proposed in [2]. However, in typical sequences the number of frames where registration is successful is a small percentage of the total number of frames. We extend the above algorithm to deal with those frames that have only been partially segmented. We use the temporal relationships between articulated chains to register them to body segments in as many of the remainder of the frames as possible. These segmented articulated chains serve as shape cues in the tracking algorithm as well.

The block diagram of the tracking algorithm is presented in Fig. 1. We initialize our tracking algorithm with a frame that has been completely segmented and registered. We propose a two-
Fig. 1. The schematic of the tracking algorithm showing the different cues used in the tracking.

part tracking algorithm that uses both motion and shape cues and consists of a predictor and corrector. The tracking algorithm is as follows.

- Compute 2D pixel displacement between frames at times $t$ and $t + 1$. Use the pixel displacement of all segments in all images to compute change in 3D pose. The predicted pose is the current pose plus the estimated pose change.
- For all articulated chains whose skeleton curves are present, perform pose correction using skeleton curves (3D shape cues).
- For articulated chains whose skeleton curves are not available, perform pose correction using 2D shape cues such as silhouettes and motion residues.

The performance of a tracking system typically increases with the number of independent observations, and to that end our system uses different kinds of cues that can be estimated from the images. We use both motion information (in the form of pixel displacements), as well as shape information (such as skeleton curves, silhouettes, and “motion residues”, hereafter referred to as shape cues). The motion and shape cues are complementary in nature and work together to alleviate the drift and local minima problem that are manifest when they are applied independently. Since we use motion and shape cues in our tracking algorithm, we are able to better deal with cases where the body segments are close to each other, such as when the
arms are by the side of the body in a typical walking posture. Purely silhouette-based methods, including those that use voxels, experience difficulties in such cases. Indeed, we use a voxel-based algorithm to initialize the pose and commence the tracking, but the registration algorithm used in the initialization fails in a host of cases where the body segments are too close to each other or when errors in the 2-D silhouette estimation cause holes and gaps in the voxel reconstruction. Silhouette or edge-based methods also have the weakness that they will not be able to deal with rotation about the axis of the body segment.

We also propose a smoothing algorithm that smooths the trunk pose (three rotational and three translational parameters) using the estimated pose. This is an optional step in the algorithm and improves the performance of our tracker. Since the trunk forms the root of the kinematic chain in our model, smoothing the trunk alone leads to a smoother estimate of the pose of the entire body. The smoothed trunk estimate is used as an input to the tracking algorithm; it is not a post processing step.

In our experiments, we use eight cameras that are placed around the subject. While the tracking algorithm performs well with four to six cameras, we need at least eight cameras to obtain reasonable voxel reconstruction results. A visual inspection of the voxel reconstruction obtained using fewer than eight cameras was found to contain “ghost” limbs in a number of frames and was, in general, of a poorer quality. The human body model parameters of the subject were estimated automatically using the method described in [2]. 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 tracking algorithm successfully tracks the pose through the entire sequence, some of which extend for more than 10 seconds. The system proposed in this paper can be used in a number of biomechanical applications, such as gait analysis as well as general human motion analysis.

We summarize related work in articulated tracking and pose estimation in Section II. We present our human body model in Section III. The corresponding pose vector is described in Section III-A. We describe the estimation of pose from the pixel displacement, and skeleton curves in Section III-B and Section III-C respectively. The complete details of the algorithm are presented in Section IV. We describe the pose initialization, temporal spline registration and
describe the two-step tracking process. We also describe the smoothing step. We present results of our algorithm on three different sequences using real images captured from eight cameras and the results are presented in Section V.

II. RELATED WORK

We address the problem of tracking articulated human motion using multiple cameras. Gavrila and Davis [3], Aggarwal and Cai [4], and Moeslund and Granum [5], provide surveys of human motion tracking and analysis methods. Sigal and Black [6] is a recent survey on human pose estimation and Wang et al. [7] provide a survey of recent developments in human motion analysis. We describe some of the related work in pose initialization and tracking using both single and multiple cameras.

A popular class of algorithms [1], [8]–[10] uses voxels in order to perform pose estimation. Chu et al. [8] use volume data to acquire and track a human body model and Cheung et al. [11] use shapes from silhouette to estimate human body kinematics. Mikić et al. [9] automatically extract the model and pose using voxel data. Mündermann et al. use Iterative Closest Point algorithm for tracking. We also use a voxel based algorithm [1] in order to perform pose initialization and model acquisition [2]. Voxel-based algorithms can be used for pose initialization in a limited number of frames in the sequence, however, there usually are a number of frames in a sequence, where errors in the voxel reconstruction (due to noise in the background silhouettes), or errors in segmentation result in missing body segments. Most voxel-based or silhouette-based algorithms typically have difficulty when body segments are close to each other. As a result, voxel-based methods cannot be reliably used for pose estimation in all the frames in a sequence.

A number of pose tracking methods use either motion cues or static cues such as silhouettes or edges as input. We first list some of the methods that estimate pose from a monocular video sequence. Wachter and Nagel [12] track persons in monocular image sequences using edge information and an IEKF framework. Sidenbladh et al. [13] provide a framework to track 3D human figures using 2D image motion and particle filters with a constrained motion model that restricts the kinds of motions that can be tracked. Ju et al. [14] use planar patches to model body segments where the motion parameters of the patches are estimated by applying
the optical flow constraint on all pixels in the predicted patches. Plänkers and Fua [15] use articulated soft objects with an articulated underlying skeleton as a model and use stereo and silhouette data for shape and motion recovery. DeMirdjian et al. [16] constrain pose vectors based on kinematic models using SVMs. Krahnstoever [17] addresses the issue of model acquisition and initialization using images from a single camera. Pose estimation from a single camera faces problems due to occlusion and kinematic singularities. Morris and Rehg [18] and Rehg et al. [19] describe ambiguities and singularities in tracking of articulated objects. While some methods [20] attempt to overcome problems of self-occlusion and others [21], [22] try to remove kinematic ambiguities in monocular pose estimation efficiently, it is, in general, preferable to use multiple cameras to perform robust and accurate tracking. More recently, there have been a number of approaches that use video from multiple calibrated cameras to estimate and track pose. Some of these methods use motion cues while others use silhouettes or edges. Yamamoto and Koshikawa [23] analyze human motion based on a robot model and Yamamoto et al. [24] track human motion using multiple cameras. Bregler and Malik [25] also use optical flow and an orthographic camera model. Gavrila and Davis [26] discuss a multi-view approach for 3D 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 [27] use silhouettes from multiple cameras to estimate 3D motion. Delamarre and Faugeras [28] use 3D articulated models for tracking with silhouettes. They use silhouette contours and apply forces to the contours obtained from the projection of the 3D model so that they move towards the silhouette contours obtained from multiple images. Moeslund and Granum [29] 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. [30], [31] use non-parametric belief propagation to track in a multi view set up. Motion based cues suffer from the problem of drifting, while silhouette based techniques are prone to converge to local minima which may be incorrect and hence it is beneficial to use multiple cues.

We present a complete initialization and tracking algorithm that uses both spatial (or structural cues) as well as motion cues to estimate and track the pose. Spatial cues are absolute and prevent
drift in the tracking, but it is not possible to extract reliable spatial cues in every frame. We therefore base our tracker on motion cues, which can be computed in every frame, using spatial cues to correct for drift. We derive a linear relation between our 3D pose vector and the pixel velocity under a perspective projection model in a multi-camera setting so that we can directly estimate the change in 3D pose from pixel displacements computed in images from multiple cameras. We also present a novel method to use spatial cues such as silhouettes and motion residues. It is also possible to incorporate other spatial cues such as edges in our method. We note that we do not constrain the motion or the pose parameters for specific types of motion (such as walking) and hence our method can be used to track arbitrary human motion and thus is quite general.

III. HUMAN BODY MODEL, POSE AND TRACKING

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), connected at joints as illustrated in Fig. 2 (c). Our model takes into account the underlying skeleton structure and flexibility of the human body model. Each rigid segment is represented by a tapered super-quadric and can be represented by a general convex 3D mesh-model. The tapered super-quadric is described in [2]. The trunk is represented using two segments in order to model the flexibility of the spine. 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). We describe in Section III-A the 3D reconstruction of the subject using the body model parameters and the pose vector. We derive the linear relation between the pixel velocity and pose velocity and describe how we estimate the change in pose from pixel displacement in Section III-B. The tracking algorithm using skeleton curves is explained in Section III-C.

A. Description of pose vector

Let $G_{4\times4}$ be a transformation matrix in homogeneous 3D coordinates consisting of a rotational component, $\omega_{3\times1}$, and a translational component, $p_{3\times1}$. The pose vector for a single body segment
comprises both components and is given by $\varphi = (p^\omega)$. $G$ is expanded as

$$
G(\varphi) = \begin{pmatrix} R & p \\ 0' & 1 \end{pmatrix} = \begin{pmatrix} e^{\hat{\omega}} & p \\ 0' & 1 \end{pmatrix}, \text{ where } \hat{\omega} = \begin{pmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{pmatrix}.
$$

We drop the dependency on pose vector for convenience in the following sections.) The articulated nature of the body is illustrated in Fig. 3. The lower trunk is referred to as the base body and is the root of the kinematic chain. It has six degrees of freedom. All body segments are attached to the base body in a kinematic chain and also have, in general, six degrees of freedom with respect to the parent segment. If the translation component of all joints is constrained to be zero, as it is in the case of most joints, it results in a pure articulated structure. It is, however, possible to allow limited translation at certain joint locations such as the shoulder (which is a “compound” joint and cannot be represented by a single rotational joint) to better model its complexity.

The pose of each body segment in the world reference frame is described by a combination of body model and pose parameters. We use the superscript $S$ to denote a structure parameter of the body and $P$ to denote a pose parameter. For e.g., $p^S$ is a joint location and is part of the body model, while $p^P$ is the translational pose at the joint and is part of the pose vector. Consider two segments, $i - 1$ and $i$ in Fig. 3, where segment $i - 1$ is the parent of segment $i$. 
Fig. 3. Articulated structure and the relative positions of body segments as a function of the body model and pose. The red, green, and blue axis set describe the pose of the coordinate frame for each segment.

Segment $i$ is connected to its parent at joint $i$, whose location is given by $\mathbf{p}^{(i)S}$ in the coordinate frame of the parent. The pose of segment $i$ is given by $G(\mathbf{p}^{(i)P}, \omega^{(i)P})$ in the coordinate frame of segment $i-1$. The complete transformation between segment $i$ with respect to segment $i-1$ is therefore given by

$$ G_{(i-1)i} = G(\mathbf{p}^{(i)S}, 0) G(\mathbf{p}^{(i)P}, \omega^{(i)P}). \quad (2) $$

$G_{ij}$ 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 base body (root of the kinematic chain with index 1) with respect to the world reference frame (index 0). The segments in Fig. 3 form a kinematic chain and the position of the $i^{th}$ segment with respect to the world coordinate frame is given by

$$ G_{0i} = G_{01} G_{12} \cdots G_{(i-1)i}. \quad (3) $$

For a strictly articulated body, $\mathbf{p}^{(i)P} = 0 \ \forall \ i > 1$. However, in order to model complex joints such as the shoulder joint, we allow $\|\mathbf{p}^{(i)S}\| < p_{\text{MAX}}$ where $i$ denotes special joints. Our human body model consists of sixteen rigid segments (as illustrated in Fig. 2). The pose of segment $i$ is given by $\mathbf{\varphi}^{(i)} = \left( \begin{array}{c} \mathbf{p}^{(i)P} \\ \omega^{(i)P} \end{array} \right)$ and the complete pose vector is given by $\mathbf{\Phi} = \left( \begin{array}{c} \mathbf{\varphi}^{(1)} \\ \vdots \\ \mathbf{\varphi}^{(16)} \end{array} \right)$. 
B. Tracking pose using pixel displacement

We consider a point on the \( i \)th segment, and its projection onto the camera image. We show that the 3D point velocity (Section III-B.1) and the corresponding 2D pixel velocity (Section III-B.2) are linear functions of the pose velocity. We then briefly explain how we estimate the change in pose using the measured pixel displacement in Section III-B.3.

1) 3D point velocity as a function of spatial velocity: We can also express the transformation, \( G \) [32], as a function of the twist vector \( \xi = \begin{pmatrix} \mathbb{P} \end{pmatrix} \) as

\[
G = e^{\hat{\xi}} = I + \hat{\xi} + \frac{1}{2!} \hat{\xi}^2 + \frac{1}{3!} \hat{\xi}^3 + \cdots, \tag{4}
\]

where

\[
\hat{\xi} = \begin{pmatrix}
0 & -\omega_3 & \omega_2 & p_1 \\
\omega_3 & 0 & -\omega_1 & p_2 \\
-\omega_2 & -\omega_1 & 0 & p_3 \\
0 & 0 & 0 & 1
\end{pmatrix}. \tag{5}
\]

We define the \( \lor \) (vee) operator to represent the 6-dimensional vector which parameterizes a twist, and its inverse operator \( \land \) (hat), so that \( \hat{\xi} \lor = \xi \). Let us consider a point \( q \), given by \( q^{(i)} \) in the coordinate frame of segment \( i \) and \( q^{(j)} \) in the coordinate frame of segment \( j \). We then have

\[
q^{(i)} = G_{ij}q^{(j)}. \tag{6}
\]

We consider the motion of frame \( j \) with respect to frame \( i \). Since we are only concerned with the instantaneous motion, we can assume that the motion is described by a constant twist, \( \xi \), so that

\[
g_{ij}(t) = g_{ij}(0)e^{\hat{\xi}t} = G_{ij}e^{\hat{\xi}t}. \tag{7}
\]

We use \( g_{ij}(t) \) to denote the transformation as a function of time and \( G_{ij} = g_{ij}(0) \) is a constant. We consider motion at \( t = 0 \) without loss of generality. Considering the velocity of the point \( q \) in frame \( i \), we have

\[
\dot{q}^{(i)} = \dot{g}_{ij}q^{(j)} + g_{ij}\dot{q}^{(j)} = \dot{g}_{ij}q^{(j)}. \tag{8}
\]

\[
= \dot{g}_{ij}q^{(j)}. \tag{9}
\]
where the second equation follows because the point is fixed in frame \( j \) and therefore \( \dot{q}^{(j)} = 0 \). Substituting (7) in (9), we get

\[
\dot{q}^{(i)}(t) = G_{ij} \hat{\xi} e^{\hat{\xi} t} q^{(j)}. \tag{10}
\]

We thus have

\[
\dot{q}^{(i)}(0) = G_{ij} \hat{\xi} e^{\hat{\xi} 0} q^{(j)} = G_{ij} \hat{\xi} q^{(j)} = G_{ij} \Upsilon(q^{(j)}) \xi, \tag{11}
\]

where

\[
\Upsilon(q) \triangleq \begin{pmatrix}
1 & 0 & 0 & 0 & -q_3 & q_2 \\
0 & 1 & 0 & q_3 & 0 & -q_1 \\
0 & 0 & 1 & -q_2 & -q_1 & 0 \\
0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}. \tag{12}
\]

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

\[
q^{(0)} = g_{0i} q^{(i)} = g_{01} g_{12} \cdots g_{(i-1)i} q^{(i)}. \tag{13}
\]

It follows that

\[
\dot{q}^{(0)} = \left( \dot{g}_{01} g_{12} \cdots g_{(i-1)i} + \dot{g}_{01} g_{12} \cdots g_{(i-1)i} + \cdots + g_{01} g_{12} \cdots \hat{g}_{(i-1)i} \right) q^{(i)} \tag{14}
\]

\[
= \left( g_{01} \hat{\xi}^{(1)} \cdot g_{12} \hat{\xi}^{(2)} \cdots g_{(i-1)i} + \cdots + g_{01} g_{12} \cdots g_{(i-1)i} \hat{\xi}^{(i)} \right) q^{(i)} \tag{15}
\]

\[
= g_{01} \hat{\xi}^{(1)} q^{(1)} + g_{02} \hat{\xi}^{(2)} q^{(2)} + \cdots + g_{0i} \hat{\xi}^{(i)} q^{(i)} \tag{16}
\]

\[
= g_{01} \Upsilon(q^{(1)}) \xi^{(1)} + g_{02} \Upsilon(q^{(2)}) \xi^{(2)} + \cdots + g_{0i} \Upsilon(q^{(i)}) \xi^{(i)} \tag{17}
\]

\[
= \begin{pmatrix}
g_{01} \Upsilon(q^{(1)}) \\
g_{02} \Upsilon(q^{(2)}) \\
\vdots \\
g_{0i} \Upsilon(q^{(i)})
\end{pmatrix} \begin{pmatrix}
\xi^{(1)} \\
\xi^{(2)} \\
\vdots \\
\xi^{(m)}
\end{pmatrix} = F(\Phi, q) \Xi \tag{18}
\]
where $\Xi = \left( \begin{array}{c} \xi^{(1)} \\ \vdots \\ \xi^{(m)} \end{array} \right)$. Let the pose at time $t = 0$ be $\Phi(0) = \left( \begin{array}{c} \varphi^{(1)(0)} \\ \vdots \\ \varphi^{(m)(0)} \end{array} \right)$ and the pose at time $t = 1$ be $\Phi(1) = \left( \begin{array}{c} \varphi^{(1)(1)} \\ \vdots \\ \varphi^{(m)(1)} \end{array} \right)$. The pose at $t = 1$ for each segment $i$ in the body is then given by

$$\varphi^{(i)(1)} = \left( \hat{\varphi}_i(1) \hat{\xi} \right)^\vee.$$  

(20)

We can represent the set of operations (20) using the abbreviated version

$$\hat{\Phi}_{t+1} = \hat{\Phi}_t \hat{\Xi},$$

(21)

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

2) 2D pixel velocity as a function of spatial velocity: We have shown in [33] that if we use a perspective projection to project the 3D point on to the camera, the resulting pixel velocity is still a linear function of the spatial velocity. Let $P_{3 \times 4} = \left( \begin{array}{c} P'_1 \\ P'_2 \\ P'_3 \end{array} \right)$ be the projection matrix, then the pixel value is given (in homogeneous coordinates) by $q^{(c)} = P q^{(0)}$. Let $u$ be the pixel coordinates, then $u_1 = q^{(c)}_1 / q^{(c)}_3$ and $u_2 = q^{(c)}_2 / q^{(c)}_3$. We then have the pixel location and velocity given by (22-23) as shown in [33]. We represent the matrix in (23) as $E(P, q)$ in (24). We thus combine (19) and (24) to express the 2D pixel velocity as a linear function of the 3D pose velocity, $\Xi$, in (25).

$$\begin{pmatrix} u_1 \\ u_2 \end{pmatrix} = \frac{1}{P'_3 q^{(0)}} \begin{pmatrix} P'_1 \\ P'_2 \end{pmatrix} q^{(0)}$$

(22)

$$\begin{pmatrix} \dot{u}_1 \\ \dot{u}_2 \end{pmatrix} = \frac{1}{P'_3 q^{(0)}} \begin{pmatrix} P'_1 \\ P'_2 \end{pmatrix} - \frac{1}{P'_3 q^{(0)}} \begin{pmatrix} (P'_1 q^{(0)}) P'_1 \\ (P'_2 q^{(0)}) P'_2 \end{pmatrix} \dot{q}^{(0)}$$

(23)

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

(24)

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

(25)

3) Estimating pose from pixel displacement: We can estimate $\Xi$ from the pixel velocity [34], so that we can use (21) 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 can compute the projection of each of these points for all the cameras as a function of the pose \( \Phi \). We call this stacked vector \( C(\Phi) \). We can also compute the pixel displacement matrix \( D(\Phi) = E(P, q)F(\Phi, q) \) and describe it as a function of \( \Phi \). \( D() \) and \( C() \) are functions of both the point coordinates and the projection matrices besides \( \Phi \), but as these are fixed for a given frame, we do not explicitly express them for the sake of simplicity. We therefore have

\[
\begin{align*}
    u &= C(\Phi) \\
    \dot{u} &= D(\Phi)\Xi.
\end{align*}
\]

The state vector in our state-space formulation is \( \Phi_t \) and the state update and observation equations are given by (28-29).

\[
\begin{align*}
    \text{State Update} : \Phi_{t+1} &= \Phi_t \hat{\Xi}_t \\
    \text{Observation} : \Delta u &= u_{t+1} - u_t \approx D(\Phi)\Xi
\end{align*}
\]

(29) follows from the first order Taylor series approximation

\[
\begin{align*}
    u_{t+1} = u_t + \dot{u}_t + \frac{1}{2} \ddot{u}_t + \cdots \approx u_t + \dot{u}_t
\end{align*}
\]

We can then use an iterative algorithm to estimate the pose from the given pixel displacement using the following steps. Let \( \Phi_t \) be the estimated pose at time \( t \). Let \( \Phi_t^0 = \Phi_t \) and \( k = 0 \).

1) Let \( k = k + 1 \).
2) Let \( \Delta u^{(k)} = \Delta u - (C(\Phi_t^{(k)}) - C(\Phi_t)) \)
3) Compute \( \Xi^{(k)} = \left( D(\Phi_t^{(k)})'D(\Phi_t^{(k)}) \right) D(\Phi_t^{(k)})'\Delta u^{(k)} \).
4) Update \( \Phi^{(k+1)}_{t+1} = \left( \Phi^{(k)}_{t+1} \hat{\Xi}^{(k)}_{t+1} \right)^\vee \).
5) Check for convergence. If converged, or \( k \) exceeds number_iterations, go to 1.
6) Set \( \Phi_{t+1} = \Phi_{t+1}^{(k+1)} \).

We have several pixel displacement measurements from multiple cameras and the estimation equation is highly over-constrained so that we can perform a least squares estimate. We also find that the multi-view constraints are able to overcome kinematic singularities and occlusions which are the bane of monocular pose estimation.
C. Tracking pose using skeleton curves

We use the method proposed in [1], to perform bottom up segmentation of the voxel data as belonging to different articulated chains and also register them. The voxel structure before and after segmentation is presented in Fig. 4 (a) and (b) respectively. We obtain a skeleton curve for each segmented articulated chain as shown in Fig. 4 (c). The corresponding skeleton model is presented in Fig. 4 (d). For a given pose vector, the skeleton model is computed as described in Section III-A. Each skeleton curve (one for each articulated chain) is represented by points which are uniformly spaced on the curve. In order to estimate how well a skeleton model fits a skeleton curve, we need to compute some kind of “distance” between the skeleton curve and skeleton model. Since the skeleton curves are all registered, we can compute the distance between each skeleton curve and its corresponding skeleton model independently. The computation of this “distance” is described in the following paragraph.

Consider a set of ordered points \( x_1, x_2, \ldots, x_n \), on a skeleton curve corresponding, for e.g., to the arm (See Fig. 5). The corresponding 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 \( L_j \). We need to assign each point to a line segment. Since the set of points on the skeleton curve is ordered, we impose the constraint that the 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 approximately equal to length of line segment \( L_2 \). For the above assignment, the distance between the skeleton curve is given by the vector \( (e_1 \ldots e_{n_1} e_{n_1+1} \ldots e_{n_2} e_{n_2+1} \ldots e_{n_2})' \). \( n_1 \) and \( n_2 \) are chosen so as to minimize the sum of the elements in the vector.

IV. ALGORITHM

We present in this section, the details of our algorithm, including preprocessing, pose initialization and pose tracking for a subject in a given sequence. The pre-processing consists of using images obtained from multiple cameras to compute silhouettes and voxels and is described in Section IV-A. We estimate the parameters of the human body model as described in [2]. We assume that we are able to perform stand-alone registration in at least one frame in the sequence using the method described in [1]. We can then initialize the pose using estimated 3D cues (skeleton curves). Typically, the pose can be initialized in several frames in the sequence. In the majority of the frames however, the stand-alone registration is unsuccessful, and we are left with unregistered skeleton curves. We propose a temporal registration scheme in which we register skeleton curves using their temporal relation. The pose initialization procedure and the temporal spline registration method is described in Section IV-B. We then describe our algorithm that tracks the pose in two steps; the prediction step using motion cues, and the correction step using 2D and 3D shape cues in Section IV-C. We also describe an optional smoothing step in Section IV-D.
Fig. 6. Processing images to compute silhouettes and voxels

A. Pre-processing

We use images obtained from $N = 8$ calibrated cameras. The calibration parameters are obtained using the LED pointer based algorithm of Svoboda [35] and a simple calibration rig to determine scale and world reference frame. We perform simple background subtraction to obtain foreground silhouettes as shown in Fig. 6. In order to compute the voxel structure, we project points on a 3D grid (in the volume of interest) to all the camera images. All points that are projected to an image coordinate that lies inside the silhouette in all the images are considered to be part of the subject. In general, we could consider all points that lie inside the silhouette in at least $N - M$ images, where $M$ could take values $0, 1, 2, \cdots$ depending on the number of cameras in use. A non-zero value of $M$ lends robustness to background subtraction errors. The voxel reconstruction results using the silhouettes in Fig. 6 (b) is presented in Fig. 6 (c). We set $M = 0$ in our experiments.

B. Pose initialization and temporal registration

We perform bottom-up segmentation of the voxel data using Laplacian Eigenmaps [1] to obtain the different articulated chains. The method maps voxels on each articulated chain to different smooth 1D curves 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 can compute the skeleton curve using smoothing splines as described in [1]. The method to initialize the pose of the subject using the registered skeleton curve is presented in Section IV-B.1.
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 possibly due to errors in voxel reconstruction or segmentation, examples of which are presented in Fig. 8. We present a temporal registration algorithm to register skeleton curves in such frames in Section IV-B.2.

1) Pose initialization: The pose is initialized for a completely registered frame using the following algorithm. The skeleton curve is sampled at regular intervals of 20mm 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 the images in Fig. 7 (d).

   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 in the direction of the line joining the right pelvic joint to the left pelvic joint. This direction is set to be the average of the rays 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. This direction is estimated using the direction of the feet and is orthogonal to the computed $yz$ plane. The $xyz$ 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. The initial estimate of the pose is illustrated in Fig. 7 (f).

2) **Temporal spline registration**: Two examples where registration of skeleton curves to articulated chains in a stand-alone frame fails are illustrated in 8. In one of the examples, the head is missing, due to errors in background subtraction, and in the other seven, instead of six, spline segments are discovered. In such cases, we can register the skeleton curves using the temporal relation between skeleton curves. Let $S^A = \{x_1^{SA}, x_2^{SA}, \ldots, x_{n_A}^{SA}\}$ and $S^B = \{x_1^{SB}, x_2^{SB}, \ldots, x_{n_B}^{SB}\}$ 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

$$d(S^A, S^B) = \frac{1}{n_A + n_B} \left( \sum_{i=1}^{n_A} \min_j (\|x_i^{SA} - x_j^{SB}\|) + \sum_{i=1}^{n_B} \min_j (\|x_i^{SB} - x_j^{SA}\|) \right).$$

Let us assume that frames at time $t_0$ and $t_1$ are registered using the spatial registration method referred to in the previous section. We need to register skeleton curves for the frames between $t_0$ and $t_1$. Let $R_i^t$ represent the reference skeleton curve for the $i^{th}$ articulated chain at time instant $t$. The following is the temporal registration algorithm. We typically set $d_{\text{THRESHOLD}} = 50\text{mm}$.

1) Set $t = t_0$.
2) Set $R_i^t = S_i^t$ for $i = 1, \ldots, 6$.
3) Set $t = t + 1$.
4) Let the skeleton curves at $t$ be $S_1^t, \ldots, S_{N_t}^t$. Note that the $N_t$ may not be equal to six.
5) For each $R_i^{t-1}$, find the closest curve, $S_i^{r_i}$, if it exists, such that $d(S_i^{r_i}, R_i^{t-1}) < d_{\text{THRESHOLD}}$ and the mapping is unique.
6) Set the new reference frame. If $R_{i-1}^t$ has a registered candidate, then set $R_i^t = S_i^r$, else set $R_i^t = R_{i-1}^t$.

7) If $t = t_1 - 1$, stop, else go to Step 3.

We use the above algorithm to perform reverse temporal registration as well, i.e., we start at $t = t_1$ and proceed backwards in time. Any skeleton curve that is not registered to the same articulated chain in the forward and backward temporal registration process is discarded.

C. Pose tracking

Our tracking algorithm consists of two steps, a prediction step and a correction step. Given the pose at $t$, we predict the pose at time $t + 1$ using motion cues. The prediction is described in Section IV-C.1. We correct the predicted pose using 2D and 3D shape cues. We described in Section IV-B.2, the temporal registration algorithm. However, some of the frames may be missing some of the articulated chains. For a given frame, and for a given articulated chain, we use the 3D articulated chain if it has been registered, or use 2D shape cues otherwise as described in Section IV-C.2.

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. We then combine the pixel displacement for a set of bodies in all the images into a single matrix equation using which we estimate the change in 3D pose.

a) Pixel-body registration: We register each pixel in each image to its 3D coordinate and determine the body segment it belongs to. We can thus obtain a 2D mask for each body segment in each image and we can impose a rigid motion model for all pixels belonging to the same segment (mask). In order to determine the correspondence between a pixel and the body segments, we convert each body segment super-quadric into a triangular mesh and project it onto each image, and compute the depth at each pixel by interpolating the depths of the triangle vertices. We can thus fairly easily extend our algorithm to use triangular mesh models instead of super-quadrics. Since the depths of all pixels are known, we can resolve self-occlusions. Fig. 9 illustrates the projection of the body segments onto images from two cameras. Different colors denote different...
body segments. We can also compute 3D coordinates of the points corresponding to the pixels using interpolation.

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

$$\delta(u, \psi) = s \begin{pmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{pmatrix} (u - u_0) + \delta u,$$  \hspace{1cm} (32)$$

where $u_0$ denotes the projection of the joint location. We find that the above parametric representation is more robust than an affine model because we can set meaningful upper and lower bounds on each parameter. Let $u_1, \ldots, u_n$ be the pixels registered to a given segment and illustrated in Fig. 10 (a). We compute that value of $\psi \in [\psi_0 - \psi_B, \psi_0 + \psi_B]$ for the segment that minimizes the residue given by $e^T e$, where $\psi_B$ denotes the bounds on the motion that we impose, and the $j^{th}$ element of $e$ is given as

$$e_j = I_t(u_j) - I_{t+1}(u_j + \delta(u_j, \psi)).$$  \hspace{1cm} (33)$$

$\psi = 0$ implies no motion. Fig. 10 illustrates the pixel displacement computation. Fig. 10 (a) is the smoothed intensity image at time $t$. Fig. 10 (b) is the difference between the intensity images at time $t$ and $t + 1$. This is the same as the “motion residue” for $\psi = 0$. We note that if the actual motion of the pixel agrees with the estimated motion, then the motion residue for the pixel

![Fig. 9. Pixel registration](image1)

![Fig. 10. Pixel displacement and Motion Residue](image2)
is zero and otherwise it is some non-zero value. We note that the motion $\psi = 0$ agrees with the motion of the background pixels (the region left of the mask) which is stationary. However $\psi = 0$ does not agree with the motion of the foreground pixels. Fig. 10 (c) is the difference between images at time $t$ and $t+1$ warped according to the estimated motion and is the “motion residue” for the optimal $\psi$. We again note that the estimated motion agrees with the actual motion for the pixels in the mask, but does not agree with the motion for the background pixels. The value of the pixels in the region of the mask is close to zero where the estimated pixel displacement agrees with the actual pixel displacement. 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. Fig. 10 (d) illustrates the computed pixel displacement for pixels in the mask.

c) Pose Prediction: We need to predict the pose $\Phi_{t+1}$ given $\Phi_t$ from the pixel displacement.

The basic relationship between the pose velocity $\Xi$ and the pixel velocity and how we compute $\Phi_{t+1}$ from the pixel displacement was described in Section III-B.2-III-B.3. Since we have registered pixels to 3D points on the model, we can construct the matrix relationship between the pixel velocity and the change in pose. While our body model and our pose estimation algorithm allows rotation and translation for 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 are minimum [33]. We also estimate the pose of the subject in multiple steps, starting at the root of the kinematic chain as illustrated in Fig. 11. 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 limbs, and in the final step we estimate the pose for all the segments save the trunk segments. We allow only the base body to translate freely. We allow the shoulder joint to translate under the following constraint, where $P^{\text{SHOULDER}}$ is the translation at the shoulder joint.

$$\|P^{\text{SHOULDER}}\| \leq 20\text{mm}$$  \hspace{1cm} (34)

2) Correction using 3D and 2D shape cues: The pose can be corrected for all the articulated chains in a given frame that have been registered (spatially or temporally) using the 3D shape cues (skeleton curves). The pose of each articulated chain is corrected using the skeleton curves.
as presented in Section III-C. The pose parameter search space is bounded and centered around the pose predicted using motion cues. In the absence of 3D shape cues, we can use 2D shape cues in the form of silhouettes and motion residues. This allows us to use the framework irrespective of which spatial cues are available. The “motion residue” for the example in Section IV-C.1.a is presented in Fig. 10 (d). The “motion residue” for a given segment provides us with the region that agrees with the motion of the mask and helps us spatially delineate the segment in the image. We combine the “motion residue” and the silhouette as shown in Fig. 12 to form an error image for that segment. We now have the pixel-wise error image for each camera and a given segment as well as a mask for the body segment for the body segment for a given image as illustrated in Fig. 12 (e). For a given 3D pose, $\varphi$, of the segment we can project the axis onto
each image. The red line in Fig. 12 (e) denotes the 3D axis of the segment denoted in blue. For a new value of the pose we get a different axis (for e.g., the cyan line). The 2D motion can be represented by a displacement and rotation of this axis (we ignore the scaling). We compute the error of a 3D pose by warping the segment mask according to the mentioned displacement and rotation and summing the value of all the pixels in the error image that belong to the mask. Thus we can express the error of each 3D pose directly by stacking the error of the pixels belonging to that segment in each image. We minimize this error function in a pose parameter space that is centered around the predicted pose using non-linear optimization toolbox. We illustrate the results of the pose correction for the above example in Fig. 13. The red line represents the initial position of the axis of the body segment and the cyan line represents the new position.

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 component 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. Smoothing the pose of the root segment of the chain has an impact on the pose of all the body segments. We perform the smoothing using the pose estimated from the skeleton curves. The skeleton curve of the trunk and head is used to estimate the 3D location of the trunk, and the the presence of the limbs can be used to estimate the orientation of the trunk. The trunk location is interpolated for frames missing the trunk skeleton curve and the smoothed trunk pose estimate is computed using the cubic smoothing spline function of the MATLAB Toolbox. The translational components of the pose of the trunk for one of the test sequences is presented in Fig. 14. The translational components are given by \( p = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \).

V. 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 \( \times \) 484 at a frame rate of 30 frames per second. The external and internal
camera calibration parameters for all the cameras were obtained using Tomas Svoboda’s camera calibration algorithm [35] and a simple calibration device to compute the scale and the world reference frame. We present results for three sequences that include the subject swinging the arms in a wide arc (300 frames, 10 seconds) in Fig.16, walking in a straight line (65 frames, 2 seconds) in Fig.17, and walking in a circular path (300 frames, 10 seconds) in Fig.18. Fig.15 illustrates the motion of the base body in the world reference frame in the three sequences. Our results show that using only motion cues for tracking causes the pose estimator to lose track eventually, as we are estimating only the difference in the pose and the error accumulates. 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 body parts are successfully tracked in the three sequences.

![Sequence 1: Images from camera 1](image1)

![Sequence 1: Images from camera 3](image2)

Fig. 16. Tracking results for sequence 1

**VI. Conclusion**

We have presented a complete pose initialisation and tracking algorithm using a fairly complete and flexible human body model that allows translation at complex joints such as the shoulder joint. The human body model is estimated automatically from the sequence using the algorithm [2]. Initialisation is performed based on single frame segmentation and registration [1] of voxel data. We propose an algorithm to perform temporal registration of partially segmented voxels so that we can use partially segmented voxels for tracking. We use 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 present results on sequences with different kinds of motion and observe that the several independent cues used in the tracking enable it to perform tracking in a robust manner.
Fig. 17. Tracking results for sequence 2

REFERENCES


Sequence 3: Images from camera 1

Sequence 3: Images from camera 3

Fig. 18. Tracking results for sequence 3


