Available online at www.sciencedirect.com
ScienceDirect
Computer Vision and Image Understanding 104 (2006) 190–209
Computer Vision and Image Understanding
www.elsevier.com/locate/cviu

Markerless tracking of complex human motions from multiple views

Roland Kehl [a,*], Luc Van Gool [a,b]

[a] Computer Vision Laboratory (BIWI), ETH Zürich, Switzerland
[b] ESAT-PSI, University of Leuven, Belgium

Received 31 January 2006; accepted 25 July 2006
Available online 25 September 2006

Abstract

We present a method for markerless tracking of complex human motions from multiple camera views. In the absence of markers, the task of recovering the pose of a person during such motions is challenging and requires strong image features and robust tracking. We propose a solution which integrates multiple image cues such as edges, color information and volumetric reconstruction. We show that a combination of multiple image cues helps the tracker to overcome ambiguous situations such as limbs touching or strong occlusions of body parts. Following a model-based approach, we match an articulated body model built from superellipsoids against these image cues. Stochastic Meta Descent (SMD) optimization is used to find the pose which best matches the images. Stochastic sampling makes SMD robust against local minima and lowers the computational costs as a small set of predicted image features is sufficient for optimization. The power of SMD is demonstrated by comparing it to the commonly used Levenberg–Marquardt method. Results are shown for several challenging sequences showing complex motions and full articulation, with tracking of 24 degrees of freedom in ≈1 frame per second.

Keywords: Markerless tracking; Full body tracking; SMD; Multi-view tracking; Markerless motion capture; Multiple cues; Volumetric reconstruction

1. Introduction

Over the last years, the tracking of articulated structures such as human bodies has gained in popularity. Applications include surveillance, human–computer interaction and computer based animations in games and the movie industry.

These applications require the solution of a common task: the one of estimating the body pose from observed images. This task of digitally recording the motions of a human is known as *Motion Capture*. Motion capture records the pose of a person on each time instant, thus recording his/hers motions over time. Since the human body consists of several rigidly moving body parts the pose of a person can be seen as a set of parameters which describe the actual placement of these parts.

Commercial vision-based motion capture systems require the actor to wear a black costume with white, clearly outstanding dots as markers on it. However, markers and the corresponding special clothing are not always desirable or even applicable.

This work presents a model-based method for markerless tracking of the full body pose from multiple camera views. When omitting the markers, inferring the body pose from video sequences becomes difficult as the correlation between the observations and the pose is less immediate. Moreover, the high dimensionality of the state space and the many ambiguities caused by occlusion and fast motions require rather sophisticated tracking and expensive matching between the predicted pose and the acquired images. However, important progress has been made over the last years as discussed in Section 2.

A description of our multi-camera setup can be found in Section 3. In Section 4 we describe the set of image cues we use for tracking. We present a fast voxel-based procedure to compute a volumetric reconstruction of the person for tracking the model's surface. As the volumetric reconstruction

* Corresponding author. Fax: +41 44 63 21199.
E-mail addresses: rkehl@vision.ee.ethz.ch (R. Kehl), vangool@vision.ee.ethz.ch (L.V. Gool).

1077-3142/$ - see front matter © 2006 Elsevier Inc. All rights reserved.
doi:10.1016/j.cviu.2006.07.010

is inaccurate at times, the tracking is further strengthened by matching model contours against image edges. We propose to use RGB color edges instead of grayscale ones to avoid clutter edges caused by shadows or wrinkles on the clothing. Color is also considered for tracking in terms of a 3D color segmentation for the surface matching.

In Section 5 we introduce our articulated body model. We use superellipsoids to define its surface which approximates the human shape reasonably well while still being compact in its description.

Our optimization framework is presented in Section 6. First, a robust objective function is built from the multiple cues. Then, this objective function is optimized using *Stochastic Meta Descent* (SMD), a gradient descent with local step size adaptation. Stochastic sampling is a strong feature of SMD and particularly interesting for tracking: shuffling the set of points used for evaluating the objective function on each iteration lets SMD overcome local minima and lowers the computational costs as a smaller sample set can be used.

Finally, in Section 7, we present tracking results for several sequences where a person performs complex motions and wears different, casual clothes.

2. Related work

The problem of markerless tracking of the human pose has attracted the attention of many researchers in the past years. Approaches can be classified by the number of cameras they use, i.e. monocular or multi-view, or the image cues they exploit for the pose estimation task. Most of them have in common that they apply an articulated body model and match it against the image cues.

Monocular approaches have the advantage that they work with a simple hardware setup, i.e. one, uncalibrated camera. This makes them particularly interesting for surveillance applications where a multi-camera calibration is not applicable in most cases. At least for the moment, the advantages of multi-view systems can easily outweigh this feature, however, as problems with occlusions and appearance ambiguities can be strongly reduced. This leads to faster and more robust algorithms. Thus, fast, markerless pose tracking as proposed in this paper but from monocular sequences seems quite remote still, even if promising work has been published [30,29,3,36].

Several multi-view approaches for markerless body tracking have also been published lately. Gavrila and Davis [13] use 4 calibrated cameras. The cost function is calculated from extracted edges in the neighborhood of the target contour predictions. Then a robust variant of the Chamfer distance (the directed Chamfer distance) is computed. As their model, fleshed out by tapered quadrics, has 22 DOFs, the estimation is processed by recursive search space decomposition: using a best-first search, the best torso/head configuration is found, then the arms ... etc. Furthermore, the use of tight-fitting clothes, with sleeves of contrasting colors, makes the segmentation easier. In our case, the user can wear casual clothes, possibly with homogeneous colors.

Kakadiaris and Metaxas [17] have developed a method to acquire the shape of a human body surrounded by 3 orthogonal cameras. The body was modeled by a set of deformable shapes. Later [18], they use this representation to perform 3D body tracking. Information from image contours yield physical forces that drive the search for the state parameters. An extended Kalman filtering is employed for the model prediction. Furthermore, at each frame, an active selection among the 3 cameras allows to choose views presenting the best information in order to deal with occlusions. Tracking of a human arm against a dark background is demonstrated.

Delamarre and Faugeras [8,9] presented a method based on physical forces between the model boundaries and the observed silhouettes. They use Maxwell's demons to compute the physical forces which move the model towards the final estimation of the real pose. They apply an algorithm introduced by the robotics community to solve the dynamics of their kinematic system in real-time. However, the expensive extraction of the observed silhouettes based on geodesic active contours is jeopardizing the overall performance. The framework can handle fast movements, self-occlusions and noisy images as is demonstrated by impressive tracking results for a person running in front of three cameras.

Deutscher et al. [10] used three cameras to track a person with a modified particle filter based on a simulated annealing algorithm. Compared to the standard Condensation algorithm, the annealed particle filter (APF) reduces the number of samples and increases efficiency by a factor of up to 10. Moreover, APF localizes the tracked object even better as it increases the chances of finding the global minimum. Their body model is represented by truncated cones and the cost function takes into account edge and silhouette information. In their later work [11] they described two major improvements to the APF and reported tracking results of complex motions such as a handstand.

Carranza et al. [4] fit an articulated body model to 2D data by minimizing the overlap between the observed 2D shapes and the projections of the model. Although their method does not require explicit 3D reconstruction, an exact body model and noiseless segmentation of the person in the different videos is crucial to reach a meaningful error measurement. The tracking is done by using Powell's method and via hardware acceleration. They achieve accurate results at about 2 s per frame on distributed hardware. In [34], they enhance their silhouette-based method by incorporating texture information into the tracking process. A 3D motion field is used to refine the pose estimate.

The presented methods all exploit 2D image information for tracking. These cues only offer rather weak support to the tracker though, which quickly leads to sophisticated and therefore rather slow optimization schemes. Multiple calibrated camera allow the computation of the 3D shape of the person, a strong cue for tracking: the 3D shape only

contains information which is consistent over all the individual views wrt. some hypothesis and thus discards, for example, clutter edges or spikes in the silhouettes. The increase of the computational power offered by cheap consumer PC's allowed real-time computation of the 3D shape and created several interesting approaches to full body tracking.

Cheung et al. [5] introduced the SPOT algorithm, a rapid voxel-based method for the volumetric reconstruction of a person. Real-time tracking is achieved by assigning the voxels in the new frame to the closest body part of the previous one. Based on this registration, the position of the body-parts is updated over consecutive frames. If registration is wrong, track of the body parts can be lost. In [6] they use both color information and a shape-from-silhouette method for full body tracking, although not in real-time anymore. They use colored surface points (CSPs) to segment the hull into rigidly moving body parts, based on the results of the previous frames, and take advantage of the constraint of equal motion of parts at their coupling joints to estimate joint positions. A complex initialization sequence recovers the joint positions of an actor, which are used to track the same person in new video sequences.

Mikic et al. [21,22] proposed a similar voxel-based method for full body tracking. After volumetric reconstruction, the different body parts are located using sequential template growing and fitting. The fitting step uses the placement of the torso computed by the template growing to obtain a better starting point for the voxel labeling. Furthermore, an extended Kalman filter is used to estimate the parameters of the model given the measurements. To achieve robust tracking the method uses prior knowledge of average body part shapes and dimensions.

Mitchelson and Hilton [23] use shape and color information for tracking the full human body. In their model-based approach they use a silhouette-overlap term to overcome the need for a volumetric reconstruction. However, for model initialization, they do apply a volume carving method to retrieve the actual shape of the person's torso from a set of initialization images. An edge-proximity term and color-consistency between the model and the images further strengthen the fitness function for their hierarchical stochastic sampling scheme. They show results for simultaneous tracking of two persons.

A common problem of 3D based algorithms is the low accuracy of the reconstruction which mostly depends on the quality of the input images and the foreground segmentation. Also, localization of the body parts is often wanting because voxel-based procedures tend to result in bulky reconstructions. Adding 2D cues can increase the tracking accuracy as they offer better localization. Moreover, the robustness against erroneous 3D reconstructions can be increased in the same time. Therefore, it stands to reason then to combine 2D and 3D cues, as to get the best of both worlds.

Plänkers an Fua [25] achieved robust frontal, upper body tracking in the presence of self-occlusions by combining silhouettes and the depth information provided by three cameras. Each body part of the articulated body model is built from soft objects or metaballs which offer realistic physical deformations. Based on this model, a mathematical framework is introduced in order to compute first and second order derivatives of the cost function which is minimized by the Levenberg–Marquardt algorithm. Tracking results are reported with self-occlusions of the arms against a dark background.

Our approach combines image edges, color and a volumetric reconstruction to benefit from the advantages of both 2D and 3D cues. The volumetric reconstruction is the strongest cue and is sufficient to match the body model well against the images. However, the image edges advance the localization significantly and make the tracker more robust against erroneous 3D. Also, as the 3D is inaccurate at times, ambiguities can be solved by considering image edges and color. The optimization is performed with SMD, a stochastic method which offers the advantage that a smaller set of randomly chosen points is sufficient for tracking and therefore allows to be faster than other methods. The proposed method allows tracking of complex motions of a person from five cameras in around 1 s per frame.

3. System setup

Our camera environment allows us to activate up to 16 statically mounted IEEE1394 cameras placed all around a single working volume. The cameras are synchronized through external trigger hardware and are calibrated by the automatic self-calibration procedure proposed by Svoboda et al. [32]. For all experiments, the cameras were configured to acquire 640 times 480 images at 15 Hz. We used five cameras from well-separated directions: four look more or less horizontally and one camera provides an overhead view.

As a first step after acquisition, all images undergo a foreground/background segmentation. We use the GPU based method proposed by Griesser et al. [14] which is an efficient and enhanced version of the segmentation algorithm proposed by Mester et al. [24]. Input images and computed foreground masks for our five camera views can be seen in Fig. 1.

4. Feature extraction

Good image features give a compact and preferably complete representation of all task-relevant information in the image(s). As we are looking for a human pose, image features which give a clue about the person's pose or shape are of particular interest. The system setup yields the acquired color images and a binary shape mask of the person. Our features will be derived from these two sources.

In Section 4.1, we propose a fast method to reconstruct the 3D surface of the person. The algorithm first computes the visual hull using a voxel-based procedure and then

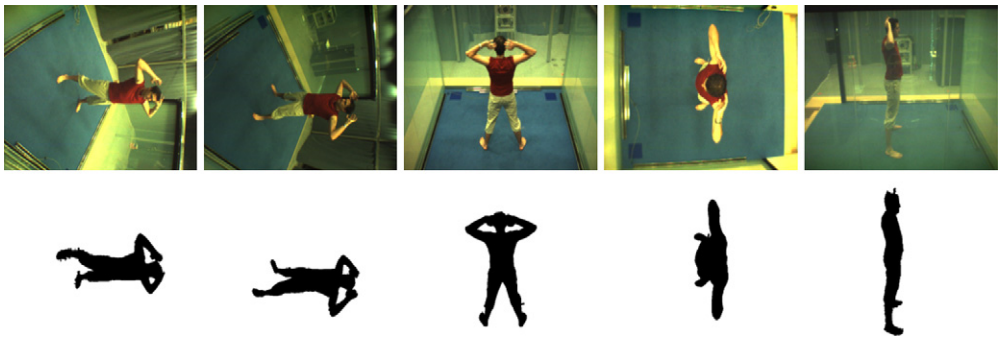


Fig. 1. The first row shows the five camera views we use for tracking and the second row the foreground-masks generated by our segmentation method.

extracts the colored surface using an efficient single-pass algorithm. Section 4.2 introduces a method to extract edges from the RGB input images. We show that a color edge detector (instead of a grayscale one) produces fewer clutter edges which are mainly caused by cast shadows or wrinkles on the clothes.

4.1. Visual hull

Methods to compute the visual hull of an object use its silhouettes in a number of available images together with the center of projections of the corresponding cameras. The intersection of the resulting projection cones define a volume which bounds the object.

Voxel based visual hull descriptions are popular but tend to be expensive as a high number of voxels have to be projected into the camera images. Most implementations speed up this process by using an octree representation to compute the result from coarser to finer resolutions [33], while others exploit hardware acceleration [16]. Our method addresses the problem the other way around. Instead of projecting the voxels into the camera views at each frame, we keep a fixed look-up table (LUT) for each camera view and store a list at each pixel with pointers to all voxels that project onto that particular pixel as illustrated in Fig. 2. This way, the image coordinates of the voxels have neither to be computed during runtime nor to be stored in memory. Instead, the LUT's are computed once at startup.

The proposed reversal of the projection allows a compact representation of the voxels: each voxel is represented by a bit mask where each bit b_i is 1 if its projection lies in the foreground of camera i and 0 otherwise. Thus, a voxel belongs to the object (i.e. is labeled as *active*) if its bitmask only contains 1's. This can be evaluated rapidly by byte comparisons.

Another advantage of our method is that the voxel space can be updated instead of being computed from scratch for each frame. A voxel only changes its label if one of the pixels it is projected to changes from foreground to background or vice versa. Therefore, as we can directly map from image pixels to voxels, we only have to look up the voxels linked to pixels which have changed their

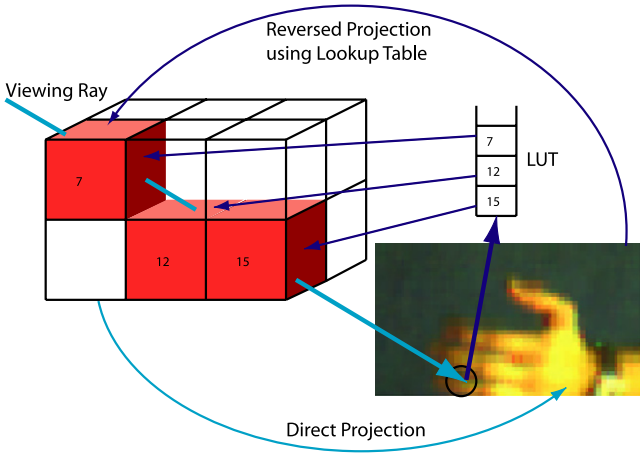


Fig. 2. A lookup-table (LUT) is stored at each pixel in the image with pointers to all voxels that project onto that particular pixel. This way, expensive projections of voxels can be avoided and the algorithm can take advantage of small changes in the images by only addressing voxels whose pixel has changed.

foreground-background status. This leads to far fewer voxel look-ups compared to standard methods where for each frame all voxels have to be visited in order to determine their labels.

The reconstruction itself is done by going pixel by pixel through all segmented (binary) images. If a pixel of the current view i has changed its value compared to the previous frame, the corresponding bit b_i for all voxels contained in the reference list of this pixel is set to the new value and their labels are determined again.

Voxels inside the object are not interesting as we want to match the model surface against the object surface. A simple but effective way to test for surface voxels is to check if at least one of their six nearest neighbors does *not* belong to the object. A noteworthy benefit of the surface extraction is the reduction of voxels which have to be taken into account for the model-observation matching. In case of a 64^3 resolution, the number of voxels on the surface is between 1100 and 1300, in case of a 128^3 resolution between 4500 and 5500.

Texturing the surface is the next logical step. In [19] a method is described which assigns a meaningful color to each surface voxel at the same time as they are identified.

Table 1
Processing times for volumetric reconstruction

No. views	Reconstruction (ms)		Texturing (ms)		Total (ms)	
	64^3	128^3	64^3	128^3	64^3	128^3
4	6.63	38.80	6.07	29.28	12.70	68.07
5	7.56	40.72	6.40	29.06	13.96	69.78
9	12.67	64.78	8.18	32.61	20.85	97.39

The table shows that processing time scales better than linearly with the number of contributing views. Our method is capable to compute a textured volumetric reconstruction at sufficient framerate for tracking even at a resolution of 128^3.

Texturing is not used for visualization purposes but rather to overcome difficult situations caused by inaccurate 3D reconstruction or by body parts making contact. Assigning a representative color to each surface voxel can improve the tracking as described in Section 6.1.3.

Table 1 shows average processing times on a PIV 3 GHz for 64^3 and 128^3 resolution. Results of our reconstruction algorithm for 64^3 and 128^3 resolution can be seen in Fig. 3.

4.2. Image edges

Image edges are broadly used for tracking as they offer good localization for model matching. Gavrila and Davis [13] use a similarity-measure based on the directed Chamfer distance between predicted model edges and observed image edges. Then, this similarity-measure is optimized with an exhaustive *best-fit* search within the *22*-dimensional search space. Wachter and Nagel [36] directly match projected gradients of the model's contours against scene edge gradients. They also include region information for better stability.

Obviously, tracking is more robust if only correct image edges are considered. The ideal case would be that the extracted edge map would contain the contours of all body parts, similar to the one we compute from the model. The silhouette extracted from the segmented foreground as shown in Fig. 4b) is therefore not sufficient. In this example the right arm can be hardly seen in the silhouette. Edge detectors can be used to extract this additional, structurally important edges quite well. However, they also produce unwanted clutter edges caused by shadows or wrinkles/texture of the clothing. In order to reduce clutter edges we perform edge detection in the RGB color space.

The use of color in edge detection increases the amount of information needed for processing which complicates the definition of the problem. For grayscale images, edges are typically modeled as brightness discontinuities. For color, the computation of a meaningful gradient is difficult.

Fig. 3. The first two images show a reconstruction computed from five cameras at resolution 64^3. The last two images show the same reconstruction at 128^3 resolution.

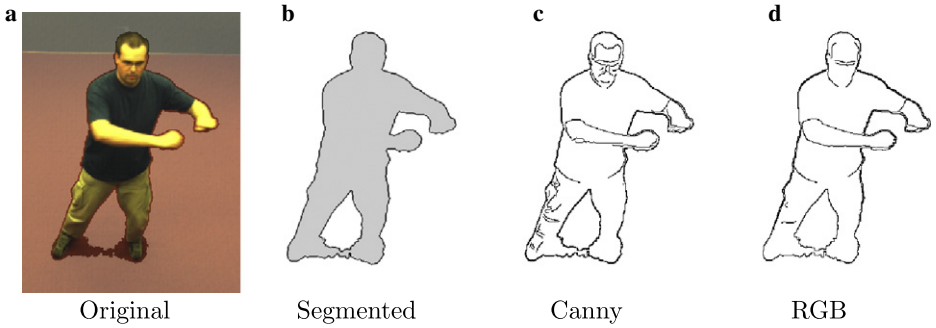


Fig. 4. Image (a) was used to test the edge detectors. The foreground mask (light grey) and the extracted silhouette of the person are shown in (b). In this example the right arm can be hardly seen in the silhouette. Image (c) shows the edge map generated with the grayscale Canny edge detector for the input image. Especially on the right leg shadow edges appear. Image (d) shows the edges extracted using the RGB edge detection which preserved all structurally important edges while suppressing those caused by shadows on the legs. Furthermore, the boundaries of the right lower arm are detected more precise by the color version. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

Wesolkowski and Jernigan [37] evaluated different color vector difference metrics and applied them to different edge detectors. They pointed out that Euclidean distance-based edge maps were much more cluttered than either the vector angle-based or combined distance measure-based results.

Therefore, following this comparison, we perform edge detection in the RGB color space by using a mixture of a Euclidean Distance (ED) between color vectors and the Vector Angle (VA) they subtend, where the weights between these criteria depends on the intensity. The use of intensity as a trade-off between these two metrics is the logical choice given that the VA metric breaks down for low values of intensity.

For two pixels, a smooth weighting function is defined that gives more weight to the VA metric for high intensities or to ED for low intensities. We approximate the intensity of a color triplet in RGB space by the average of the RGB components. Since every time two pixels are considered, the VA metric should be used only if both points have high intensities, otherwise the ED metric should be used. More details can be found in Wesolkowski and Jernigan [37].

A Difference Vector Edge Detector [35,38] is used to compute the final color gradients. This detector is a 3×3 operator that calculates the maximum gradient across the central pixel. Therefore, *4* pairs of pixels, each symmetrically arranged around the pixel where the analysis takes place, yield *4* possible 'gradients'. The one with the maximum value is considered to be the RGB gradient.

Fig. 4 shows a comparison between our RGB edge detector and the grayscale Canny edge detector. The thresholds and other parameters were individually tuned by hand for each camera.

5. Modeling a human body

In common with the majority of markerless full body tracking approaches, we use an articulated body model and fit it to data extracted from the camera images. Such a model should support the tracking of a broad variety of motions/poses while being adaptable to different human shapes. At the same time, the parameter set for describing the pose should be kept small as each additional parameter increases the dimensionality of the problem. We present a relatively accurate body model for tracking built from superellipsoids and driven by a simplified version of a human skeleton.

5.1. Superellipsoid body model

A concise mathematical description of the human pose is essential for fast tracking. A simplified skeleton, often called a "stick figure", is used to articulate the body model from a small set of parameters. These parameters describe the joint angles and shape parameters. Our simplified skeleton discards fine structures such as the hands or the feet. Their recognition from camera images is hard due to the limited resolution and the complexity of their anatomy.

Our simplified version of the human skeleton consists of 10 joints with a total of 24 degrees of freedom (DOFs) as shown in Fig. 5a). The shoulders and hips are modeled as ball & socket-joints with three angles for pitch, roll and yaw, the knees and elbows as hinge-joints with one angle for their flexion. The neck has been given only pitch and yaw angles as the twist of the head is hard to track. The torso is parameterized with all 6 DOFs of rigid motion (three translational components and three rotation angles). Together these parameters build a 24-dimensional configuration vector $\mathbf{p}$ which determines the pose of the body model.

The skeleton must have a surface to match it to our features, i.e. the volumetric reconstruction and the image edges. Different ways for approximating the surface of a human body were proposed in the literature. Models built from ellipsoids, cylinders or cones are widely used for the tracking of human bodies [3,36,23,7]. Such models tend to be rather coarse, e.g. for complex body parts such as the torso. Therefore, it is often required to combine multiple such shapes for a single body part.

Superellispoids are a special case of superquadrics [1] and offer a better approximation for complex body parts. Superellipsoids allow for a compact representation of a

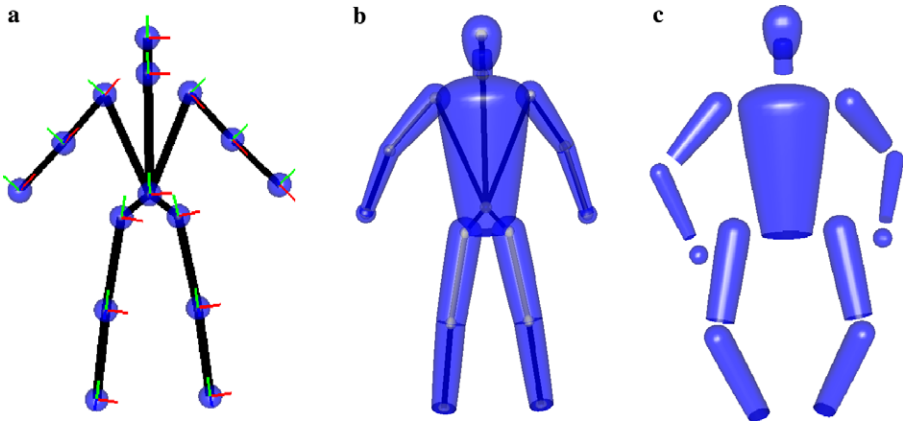


Fig. 5. The skeletal structure (a) drives the body model for tracking. The model's surface (b) is built from superellipsoids. The explosion view (c) shows that individual body parts are built from truncated superellipsoids rounded off by a spherical cap on one side.

wide variety of convex shapes, such as spheres, cylinders and cubes with rounded edges, ellipsoids of course, etc. Not surprisingly, the use of superellipsoids (or superquadrics in general) for human body modeling is not novel [20,13,30,26]. Some of them also allow for deformations such as tapering or bending to further increase the variability of the generated shapes.

Point and triangle meshes are mainly used in computer graphics where a realistic rendering is desirable. They are suited for realistic modeling of highly complex shapes at the expense of a large description, e.g. each vertex of the mesh is explicitly defined. Sand et al. [27] propose to use deformable primitives based on generalized cylinders. Each vertex is seen as a needle pointing outwards of the bone. The endpoints of the needles are then specified by an arbitrary function which defines the (deformable) shape of the body part.

In contrast to computer graphics and animation, where a body model needs to be as realistic as possible, a model for tracking can be a coarser approximation. As the features are noisy anyway, the accuracy of the model can be reduced for the benefit of a smaller set of parameters. Our human body model is built from superellipsoids. Their flexibility was used to build a model that approximates body shapes reasonably well, while still being very compact in terms of the parameters needed for its specification. Thereby, separate limbs are constructed as a seamless combination of a truncated superellipsoid and a spherical cap to round it off. Fig. 5b and c illustrate the model used for tracking.

5.2. Computations on superellipsoids

A superellipsoid is obtained by crossing two orthogonal superellipses, thus its shape is controlled by the two shape parameters $0.0 < \epsilon_1, \epsilon_2 < 2.0$ of the two superellipses. Three scaling parameters a_1, a_2, a_3 for each axis define the size. The surface of a superellipsoid is parametrized by the longitude $-\frac{\pi}{2} \leqslant \eta \leqslant \frac{\pi}{2}$ and the latitude $-\pi \leqslant \omega \leqslant \pi$. A point $\vec{r}(\eta, \omega)$ on the surface is given by the explicit equation

$$\mathbf{r}(\eta,\omega) = \begin{bmatrix} a_1 \cos^{\epsilon_1}(\eta)\cos^{\epsilon_2}(\omega) \\ a_2 \cos^{\epsilon_1}(\eta)\sin^{\epsilon_2}(\omega) \\ a_3 \sin^{\epsilon_1}(\eta) \end{bmatrix} \quad (1)$$

The surface normal directions can be obtained by

$$\mathbf{n}_d(\eta,\omega) = \begin{bmatrix} \frac{1}{a_1} \cos^{2-\epsilon_1}(\eta)\cos^{2-\epsilon_2}(\omega) \\ \frac{1}{a_2} \cos^{2-\epsilon_1}(\eta)\sin^{2-\epsilon_2}(\omega) \\ \frac{1}{a_3} \sin^{2-\epsilon_1}(\eta) \end{bmatrix} \quad (2)$$

During tracking we want to align the model with the image edges and the 3D reconstruction. Therefore, three operations on the superellipsoids surface are needed for the tracking step:

(1) Selection of random surface points for the surface matching.
(2) Identification of points lying on the superellipsoids occluding contours in the camera views, used for matching the model contours to the image edges.
(3) Identification of pairs of contour points being symmetric to the limb axis (for the symmetry corrected error measure presented in Section 6.1.2).

Selecting a set of points on the surface for tracking seems trivial as any random point on the surface may be chosen. However, uniform sampling of η and ω results in a trigonometric sampling of the surface, thus more points will be chosen on more curved regions.

In our approach, a regularly sampled grid of surface points approximates the surface where the sampling has to be done only once as a preprocessing step. We regularly sample the points along circles of latitude and meridians, as illustrated in Fig. 6 on the right plot. During tracking, random points are chosen among the sampled surface points. Furthermore, points which are hard to track, such as the ones in the armpits, are discarded.

For operations (2) and (3) we need to be able to compute the model's occluding contour efficiently. The rim of any superellipsoid—which will partly contribute to the model contours—can be obtained by solving the equation

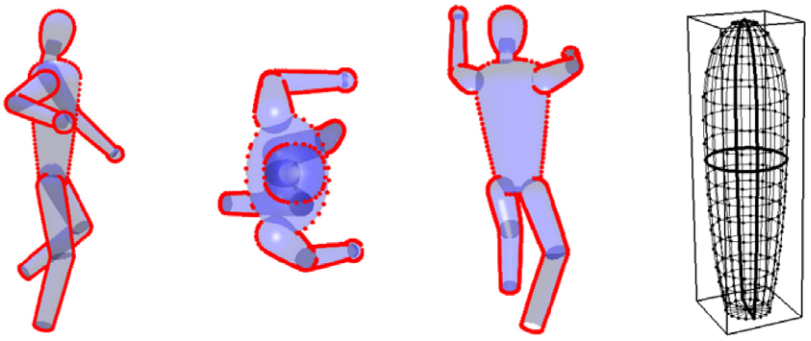


Fig. 6. The occluding contour of our model shown from three different views. The right image shows then sampling of our body parts and the bounding box of the primitive. One meridian and one circle of latitude are plotted in bold.

$$\mathbf{n}_d(\eta,\omega)\cdot\mathbf{v}(\eta,\omega)=0 \tag{3}$$

for the dot product between the direction of the viewing ray $\mathbf{v}(\eta,\omega)$ (in the superellipsoid local coordinate system) through a point $\mathbf{r}(\eta,\omega)$ with surface normal direction $\mathbf{n}_d(\eta,\omega)$. The viewing ray in local coordinates can be computed by moving the camera instead of the superellipsoid: if P is a camera's projection matrix and M represents the superellipsoid's motion, we substitute $\tilde{P}=PM=K\tilde{R}[I|-\tilde{\mathbf{C}}]$, where K denotes the unchanged calibration matrix, $\tilde{R}$ the orientation and $\tilde{\mathbf{C}}$ the center of the moved camera matrix. Therefore, if $\mathbf{r}(\eta,\omega)$ is a point on the superellipsoid surface in local coordinates, the direction of the viewing ray through this point in local coordinates becomes [15]:

$$\mathbf{v}(\eta,\omega)=\tilde{R}^{-1}K^{-1}\tilde{P}\mathbf{r}(\eta,\omega)=D\mathbf{r}(\eta,\omega) \tag{4}$$

A closed-form solution for Eq. (3) can be obtained by assuming orthographic projection, but there is too much perspective distortion in the images of cameras close to the object, like in the overhead camera most of the time. An approximation can be achieved by considering the sign change on contour locations: the rim divides the surface into a visible (Eq. (3) is <0) and invisible (Eq. (3) is >0) part. Therefore, contour points are identified by stepping through strips of surface points and checking Eq. (3) for a sign change of the dot product. We arrange the sampled surface points as strips corresponding to meridians and circles of latitude to assure that a sign change will happen on each strip. This strips are illustrated in the wireframe model in Fig. 6.

The final coordinates of the contour points can the be computed by considering the angular coordinates $\mathbf{u}_1=[\eta_1,\omega_1]^{\mathrm{T}}$ and $\mathbf{u}_2=[\eta_2,\omega_2]^{\mathrm{T}}$ and the dot products t_1 and t_2 of the two points where the sign has changed in between:

$$\mathbf{u}_c=\begin{bmatrix}\eta_c\\ \omega_c\end{bmatrix}=\frac{\frac{\mathbf{u}_1}{t_1}+\frac{\mathbf{u}_2}{t_2}}{\frac{1}{t_1}+\frac{1}{t_2}}=\frac{\mathbf{u}_1t_2+\mathbf{u}_2t_1}{t_1+t_2} \tag{5}$$

Examples of our occluding contour are shown in Fig. 6. Occlusion was handled using the procedure proposed in Section 5.3.

In Section 6.1.2 we will derive an error measurement which considers pairs of contour points which are symmetrical to the limb axis. Such a symmetry corrected error measurement increases the chance of assigning correct edges. Therefore, for operation (3), pairs of contour points symmetric to the limb axis must be identified. Such pairs can be found by only considering the strips corresponding to the circles of latitude: two contour points found on the same circle of latitude are symmetric to the main axis of the superellipsoid. Consequently, as the main axes of our superellipsoids are aligned with the limb axes, the two contour points are also symmetric to the corresponding limb axis.

5.3. Occlusion handling

A mechanism for detecting occlusions is needed to compute the visible parts of the model contours. The algorithm of Section 5.2 returns for a superellipsoid $\mathscr{E}_1$ a set of contour points $\mathbf{r}_{\mathscr{E}_1}(\eta,\omega)$. To detect if a contour point is occluded by a superellipsoid $\mathscr{E}_2$ one must compute the intersection points between the viewing ray

$$\mathbf{l}_{\mathscr{E}_1}(\eta,\omega,\lambda)=\mathbf{r}_{\mathscr{E}_1}(\eta,\omega)+\lambda\mathbf{v}_{\mathscr{E}_1}(\eta,\omega) \tag{6}$$

and the surface of $\mathscr{E}_2$. This can be done considering the *inside–outside* function $F(x,y,z)$: it returns <1 for points inside, >1 for points outside and *1* for all points on the surface of the superellipsoid. Therefore, intersection points can be found by solving $F_{\mathscr{E}_2}(x,y,z)=1,\{x,y,z\}\in\mathbf{l}_{\mathscr{E}_1}(\eta,\omega,\lambda)$. It is convenient to transform the viewing ray to the local coordinate system of $\mathscr{E}_2$. If M_1 is the transformation matrix of $\mathscr{E}_1$ and M_2 the one of $\mathscr{E}_2$, the expression of the viewing ray becomes

$$\mathbf{l}_{\mathscr{E}_2}(\eta,\omega,\lambda)=M_2^{-1}M_1\mathbf{r}_{\mathscr{E}_1}(\eta,\omega)+\lambda rot(M_2^{-1}M_1)\mathbf{v}_{\mathscr{E}_1}(\eta,\omega) \tag{7}$$

where $rot()$ denotes the rotation matrix of the transformation.

Since no closed-form solution for the intersection of a line with a superellipsoid is known, testing the inside–outside function along the line cannot be avoided. At least, first computing the intersection points of $\mathbf{l}_{\mathscr{E}_2}(\eta,\omega,\lambda)$ with the bounding box of $\mathscr{E}_2$ shortens the search significantly. Furthermore, the case if $\mathbf{l}_{\mathscr{E}_2}(\eta,\omega,\lambda)$ does not intersect $\mathscr{E}_2$ and thus no occlusion occurs can be detected in the same operation. Another trivial case, if $\mathbf{r}_{\mathscr{E}_2}(\eta,\omega)$ itself lies inside $\mathscr{E}_2$, can be detected by a single evaluation of the inside-outside function.

For all other cases, the viewing ray has to be sampled between the intersection points and tested for points with $F(x,y,z)\leqslant 0$. Doing this in a recursive way starting from the middle of the intersection points, where chances of hitting the superellipsoid are highest, this tasks becomes computationally reasonable.

6. Optimization framework

In Section 4 we discussed how we compute a volumetric reconstruction of the person and extract structurally important edges from the camera images. These correspond to our *observations*. On the other hand, there is the human body model which we can control via the configuration vector $\mathbf{p}$. For each such configuration, we can compute the occluding contours of the model in all image planes and select a set of points on the model's surface. These contours and points correspond to the image cues we expect to observe if $\mathbf{p}$ is the correct pose. They correspond to our *predictions*.

Optimizing the configuration vector $\mathbf{p}$ should bring the predictions into agreement with the observations. The objective function is described in Section 6.1. The Stochastic

Meta Descent optimization described in Section 6.2 searches for the optimal configuration.

6.1. Objective function

We need an objective function which determines how well the model contours agree with the edges found in the images and how close the model's surface (represented by a set of points) is to the reconstructed one.

Hence, our objective function $f(\mathbf{p})$ consists of two terms, one for the surface match $f_s(\mathbf{p})$ and one for the edge match $f_e(\mathbf{p})$. The two terms are computed as the sum of the errors between each predicted feature $\mathbf{x}$ and the assigned observed feature $\bar{\mathbf{x}}$.

$$\begin{aligned} f(\mathbf{p}) &= w_s f_s(\mathbf{p}) + w_e f_e(\mathbf{p}) \\ &= w_s \sum_{N_s} \delta_s(\mathbf{x}, \bar{\mathbf{x}}) + w_e \sum_{N_e} \delta_e(\mathbf{x}, \bar{\mathbf{x}}) \end{aligned} \tag{8}$$

The scaling factors w_s and w_e balance the influence of N_s surface points against that of N_e contour points.

Each error term $\delta(\mathbf{x}, \bar{\mathbf{x}})$ is based on the squared distance between a predicted feature and the assigned observed feature. However, wrong assignments produce outliers whose influence has to be limited. Therefore, we use the Leclerc error potential $\rho(z) = 1 - e^{\frac{-z}{\upsilon}}$, where $e(\mathbf{x}, \bar{\mathbf{x}})$ is the squared distance computed for individual features:

$$\delta(\mathbf{x}, \bar{\mathbf{x}}) = \rho(e(\mathbf{x}, \bar{\mathbf{x}})) \tag{9}$$

The parameter υ controls the interval that is considered normal for the squared distance $e(\mathbf{x}, \bar{\mathbf{x}})$. Using an error potential is also beneficial regarding missing correspondences due to occlusion: if a contour point is occluded, it contributes with $\delta(\mathbf{x}, \emptyset) = 1$ to the objective function, the upper limit for the error terms.

6.1.1. Surface alignment

We want to measure the distance between the model's surface and the reconstructed one where each surface is represented by a set of points. The procedure described in Section 5.2 is used to choose a set of points $\mathscr{T}$ on the model's surface. The observed surface is described by the set of voxels $\mathscr{V}$ computed according to the algorithm in Section 4.1. The set $\mathscr{T}$ is assumed to be smaller than the set $\mathscr{V}$.

An obvious measurement is the summed distance between points on the model's surface and the voxels of the reconstruction closest to them. This requires a nearest-neighbor search on all voxels of the reconstruction for each point on the model's surface, which is computationally expensive. However, by using stochastic optimization as proposed in Section 6.2, the set of points used for tracking can be rather small while preserving the optimization's robustness.

Given that for each predicted surface point $\mathbf{x} \in \mathscr{T}$ its nearest surface voxel $\bar{\mathbf{x}} \in \mathscr{V}$ is found, the squared distance becomes

$$e_s(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{2}\left(\frac{\|\mathbf{x} - \bar{\mathbf{x}}\|}{\sigma}\right)^2 \tag{10}$$

The scaling factor σ corresponds to the limb diameter at point $\mathbf{x}$ of the model. Giving the distance in units of the limb diameter makes the error distances comparable between the different cues and ensures scale-invariance regarding limbs of different dimensions.

6.1.2. Edge alignment

Given a model configuration vector $\mathbf{p}$ and the projection matrix P of a contributing view, the algorithm described in Section 5.2 returns a set $\mathscr{C}$ of contour point coordinates $\mathbf{x}$ with their normal directions $\mathbf{x}_n$ in the image plane. By using the occlusion test described in Section 5.3, the occluded contour points can be identified and considered as not assignable. Moreover, for each view the total number of occluded contour points is known and can be used to discard views where many occlusions occur. These views should be discarded as they offer weak support to the objective function. We propose to use the three views with the smallest number of occluded contour points.

For each of these three views we want to measure the alignment between the projected model contours and the edges in the corresponding camera image. An obvious solution for a distance metric would be to find for each contour point the closest edge point by following its normal direction. This entails too large a risk to fit the wrong edges, however. By preferring edge points with high image gradient magnitude and/or a gradient direction parallel to the contour normal wrong matches can be avoided. Thus, for an edge pixel $\bar{\mathbf{x}}$ with its gradient vector $\bar{\mathbf{x}}_n$, a gradient score can be computed

$$s(\mathbf{x}_n, \bar{\mathbf{x}}_n) = \|\bar{\mathbf{x}}_n\| \left[\frac{\bar{\mathbf{x}}_n}{\|\bar{\mathbf{x}}_n\|} \cdot \frac{\mathbf{x}_n}{\|\mathbf{x}_n\|}\right] \tag{11}$$

which returns a high value if the gradient magnitude is high and/or the dot product between the gradient direction and the contour normal $\mathbf{x}_n$ is high.

Moreover, the objective function should take advantage of the symmetry of the limbs since matching a symmetric pair of points increases the chance of assigning correct edges. Sminchisescu [31] showed that one can derive a symmetry corrected objective function based on the deviation between the model predicted symmetries and the matched ones. In our case, we use the gradient score (11) to strengthen the symmetry-sensitive objective function.

According to Section 5.2 pairs of contour points $\mathbf{x}^a, \mathbf{x}^b \in \mathscr{C}$ can be identified which are symmetric to the limb axis. Then, for each pair, a Bresenham line algorithm [2] is used to search along the line through $\mathbf{x}^a$ and $\mathbf{x}^b$ (plus some additional length) for edge points $\mathscr{E}$. From these edge points, a pair $\bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b$ is chosen which minimizes the length difference, scaled by the gradient scores.

$$\bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b = \arg \min_{\bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b \in \mathscr{E}} \frac{|\|\bar{\mathbf{x}}^a - \bar{\mathbf{x}}^b\| - \|\mathbf{x}^a - \mathbf{x}^b\||}{s(\mathbf{x}_n^a, \bar{\mathbf{x}}_n^a) s(\mathbf{x}_n^b, \bar{\mathbf{x}}_n^b)} \tag{12}$$

Once two edge points have been assigned, the final error for one pair is computed as the squared distance between the midpoints as

$$e_e(\mathbf{x}^a, \mathbf{x}^b, \bar{\mathbf{x}}^a, \bar{\mathbf{x}}^b) = \frac{1}{2}\left(\frac{\| \frac{\mathbf{x}^a+\mathbf{x}^b}{2} - \frac{\bar{\mathbf{x}}^a+\bar{\mathbf{x}}^b}{2} \|}{\sigma}\right)^2 \quad (13)$$

By taking the limb diameter $\sigma = \|\mathbf{x}^a - \mathbf{x}^b\|$ into account, the objective function also becomes scale invariant between the different views.

For all other contour points which do not contribute to such a symmetric pair a similar distance measurement can be obtained by simply assigning the edge point with the highest gradient score along the search line:

$$\bar{\mathbf{x}} = \arg\max_{\bar{\mathbf{x}} \in \mathscr{E}} s(\mathbf{x}_n, \bar{\mathbf{x}}_n) \quad (14)$$

Then, the distance error can be computed as the squared distance between the contour point $\mathbf{x}$ and the assigned edge point $\bar{\mathbf{x}}$, again scaled by the limb diameter σ and consistent with the handling of surface points:

$$e_e(\mathbf{x}, \bar{\mathbf{x}}) = \frac{1}{2}\left(\frac{\|\bar{\mathbf{x}} - \mathbf{x}\|}{\sigma}\right)^2 \quad (15)$$

6.1.3. Color

Color information was already used to perform foreground–background segmentation on the input images and to detect structurally important image edges. Here it is also used to split the volumetric reconstruction into regions of similar colors. Therefore, the color information does not directly contribute to the objective function $f(\mathbf{p})$ but rather confines the nearest-neighbor search for $f_s(\mathbf{p})$.

Taking into account that a point on the model's surface (and thus on the subject's surface) will not change its actual color during tracking, all voxels which have a different color can be discarded during the nearest-neighbor search for this particular point. In [19], a method was introduced which uses an adaptive probabilistic color model for each surface point on the model and uses it to segment the reconstruction in regions of similar and different colors.

The surface of the body model is divided into small patches where each patch is represented by a single color model. The color models are set up during the initialization phase: once the model is in place, we assign to each point on the model's surface the color of its nearest voxel. Then, the colors of all points which belong to the same patch are mixed together to a single color model for the patch. Each color model is represented by a Gaussian distribution in YUV space. Given the time constraints, every color channel is modeled by a single Gaussian and the color channels are assumed to be independent. During tracking, we update the mean of each Gaussian by

$$\mu_i^+ = (1-\alpha)\mu_i + \alpha c_i \quad (16)$$

with c_i the newly incoming value for color channel i and α an updating rate in the range of [0..1]. Given the noisy colors of the reconstruction and the slow capturing frequency of 15 Hz, a learning rate of 0.3 came out to work best.

The color segmentation is done before the optimization starts. Each surface voxel is tested if it fits any of the color models we have. Then, during the nearest neighbor search for the closest voxel of a surface point, those voxels can be discarded which do not fit the color-model of the point's patch. This color segmentation is illustrated for a point on the right lower arm in Fig. 7 and brings the two following advantages:

(1) The term $f_s(\mathbf{p})$ for the surface matching is not robust against body parts coming close to each other. Limbs may be left stuck to the wrong part of the data upon their actual departure. The color segmentation can help to overcome such situations if the body parts have contrasting colors. If the person wears clothes of similar colors, the segmentation has no effect, i.e. does not corrupt the tracking either.

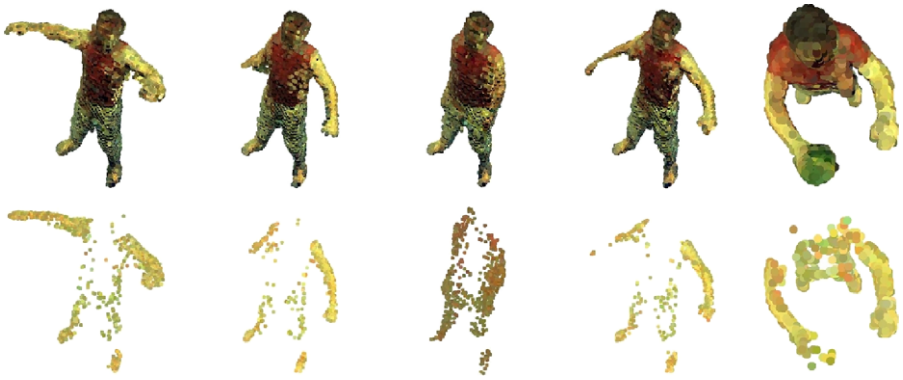


Fig. 7. 3D color segmentation for a point on the right lower arm. The first row shows the complete reconstructions, while the second row shows the voxels selected based on their color. The last column is particularly interesting, as the person holds a green cube which extends the length of the 'arm'. Color selection eliminates it. Such color-based voxel selection can also significantly speed up the tracking. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this paper.)

(2) The nearest-neighbor search for the surface matching gains in speed as voxels which do not match the color model can be discarded.

6.2. Stochastic meta descent

We propose to use *Stochastic Meta Descent* for optimization of $f(\mathbf{p})$. Stochastic Meta Descent (SMD) is a gradient descent with local step size adaptation that combines rapid convergence with excellent scalability. It is built on 3 concepts:

- The use of *stochastic sampling* of the sampling points at each iteration of SMD offers better convergence than standard optimization methods relying mostly on deterministic sampling (i.e. a fixed subset of tracking points).
- SMD performs *two levels of optimization*: firstly, it optimizes for all parameters the individual step sizes $\mathbf{a}$ at each iteration by a gradient descent which is controlled by a meta step size $\boldsymbol{\mu}$. Then, the state parameters $\mathbf{p}$ are optimized in a gradient descent manner via $\mathbf{a}$.
- SMD updates parameters while taking into account the past history of step sizes, and thus is able to capture long-range effects missed by other algorithms. This dampens erratic variations and increases the method's efficiency.

Stochastic sampling is a strong feature of SMD. Instead of using all available points for evaluating the objective function, only a subset is used and shuffled at each iteration. Indeed, randomly shuffling the sampling points lets SMD escape from local spurious minima more easily as demonstrated in Section 7.1. Furthermore, one can afford to use a smaller sample set than with other standard deterministic optimizers and can be faster as a consequence.

Here we give a concise overview of SMD. More detailed explanations can be found in [28]. If $\mathbf{g}_i$ is the gradient (of $f(\mathbf{p}_i)$) at iteration step i, the parameter vector $\mathbf{p}_i$ is updated via

$$\mathbf{p}_{i+1} = \mathbf{p}_i - \mathbf{a}_i \otimes \mathbf{g}_i, \tag{17}$$

where $\otimes$ denotes the Hadamard (i.e. component-wise) product. The vector $\mathbf{a}$ of local step sizes is in effect a diagonal conditioner for the gradient system.

The local step size vector $\mathbf{a}$ is adapted via

$$\mathbf{a}_i = \mathbf{a}_{i-1} \otimes \max\left(\frac{1}{2}, 1 + \boldsymbol{\mu} \cdot \mathbf{v}_i \otimes \mathbf{g}_i\right), \tag{18}$$

where $\mathbf{v}$ is an exponential average of the effect of *all* past step sizes on the new parameter values and $\boldsymbol{\mu}$ is a vector of meta step sizes. $\mathbf{v}$ is updated via:

$$\mathbf{v}_{i+1} = \lambda \mathbf{v}_i + \mathbf{a}_i \otimes (\mathbf{g}_i - \lambda H_i \mathbf{v}_i), \tag{19}$$

where H_i denotes the Hessian (i.e., matrix of second order derivatives), or a stochastic approximation thereof, at iteration step i. The factor $0 \leqslant \lambda \leqslant 1$ governs the timescale over which long-term dependencies are taken into account.

A condition for stopping the optimization at convergence is necessary. However, finding a suited termination rule for SMD is difficult. Shuffling the points on each iteration changes the gradient at any point in the parameter space and SMD tends to oscillate around the optimum instead of converging directly into it. To handle oscillation we propose to put a tolerance on the change of the moving average over the past n states $\mathbf{p}_i$ caused by the new state $\mathbf{p}_{i+1}$. Furthermore, an upper limit for the maximum number of iterations is given.

$$\left\| \frac{1}{n} \sum_{k=i-n+1}^{i} \mathbf{p}_k - \frac{1}{n} \sum_{k=i-n+2}^{i+1} \mathbf{p}_k \right\| = \frac{1}{n} \left\| \mathbf{p}_{i-n+1} - \mathbf{p}_{i+1} \right\| \leqslant \epsilon \tag{20}$$

$$i \leqslant i_{\max} \tag{21}$$

The maximum number of iterations $i_{\max}$ is set to 40 and n to 3 in our experiments. The tolerance ϵ is set to values between $[0.1\ldots2]$ for tracking.

6.2.1. Enforcing constraints

The freedom of movement of human joints is limited by their anatomical design and by various physical constraints. For example, one can not arbitrarily elevate one's leg sidewards or yaw an arm to the side until the elbow touches the back. Considering these constraints for tracking in a high dimensional parameter space is essential. Indeed, enforcing anatomical joint constraints not only helps to avoid implausible configurations, such as limbs intersecting each other, but they also reduce the search space we have to explore.

Joint limits can be enforced elegantly in the SMD algorithm. We enforce the constraints by means of a function that after each update (17) maps the parameters back into the feasible region:

$$\mathbf{p}^c_{i+1} = \text{constrain}(\mathbf{p}_{i+1}) \tag{22}$$

Since SMD uses the gradient not only to update the parameter vector $\mathbf{p}$, but also to adjust $\mathbf{a}$ and $\mathbf{v}$, we must make these adjustments also compatible with the constraints on $\mathbf{p}$. We do this by calculating a hypothetical 'constrained' gradient $\mathbf{g}^c$ which, applied in an unconstrained setting, would cause the same parameter change that we observe after application of the constraints. In other words, we require that

$$\mathbf{p}^c_{i+1} = \mathbf{p}^c_i - \mathbf{a}_i \otimes \mathbf{g}^c_i \Rightarrow \quad \mathbf{g}^c_i = \frac{\mathbf{p}_i - \mathbf{p}^c_{i+1}}{\mathbf{a}_i} \tag{23}$$

SMD's step size adaptation machinery can then perform accurately in the constrained space by using this constrained gradient instead of the usual one in Eq. (19).

6.3. Automatic model initialization

The initialization of the model is fully automatic. The user first has to adopt an initialization pose, standing

upright with his/her arms and legs spread in the "Da Vinci"-pose.

A first estimate of the user's position can be derived from the reconstruction's centroid. Two foot patches can be detected by using a plane sweeping algorithm near the bottom of the voxel space. The same can be done from left to right and vice-versa in order to find the hand positions and from top to bottom to find the top of the head. Then, statistics of the ratios between the different limb lengths and dimensions as proposed by Dreyfuss [12] are used to complete the skeleton.

6.4. Final algorithm

In [19] we proposed a hierarchical tracking concept by first fixing the torso and then tracking the arms, legs and the head independently. This concept lowers the dimensionality of the involved optimizations significantly and allows parallelization over multiple machines. However, regarding occlusions, the articulated chains are not truly independent anymore and must be optimized simultaneously. This is taken into account by the tracker described here.

Simultaneous optimization of all DoFs is obviously slower compared to the hierarchical method due to the lost parallelization. Even worse, experiments showed that convergence speed decreased because the higher number of parameters amplifies the oscillation effect around the optimum. As a solution we propose to first optimize all 24 DoFs at a large tolerance ϵ_1 and then refine this initial estimate hierarchically with a lower tolerance ϵ_2. During the refinement, the torso is kept fixed and only the positions of the head, the arms, and the legs are optimized. This way, the model is brought in place by the first (full) optimization and then each articulated chain is independently refined. This optimization-refinement scheme brings two advantages: first, convergence is fast for all optimizations involved and, in the refinement step, parallelization is possible.

The final optimization algorithm is summarized in Fig. 8.

7. Results

We evaluated our system with several sequences where a person performs complex motions. For the following experiments five cameras were used, with a resolution of 640 times 480 at 15 Hz. Each camera was attached to a dedicated client machine with a 2.4 GHz AMD Athlon and 1 GB of RAM. The foreground/background segmentation and the edge detection were performed in parallel on the corresponding client machines and then sent to the main host, a 3.0 GHz Pentium IV machine. The volumetric reconstruction and the full optimization were done on the main host, whereas the refinements of the five articulated chains were distributed over the client machines.

For each frame:

(1) Perform 3D color segmentation for each color model
(2) Optimize full pose with tolerance ϵ_1
(3) Optimize five articulated chains independently with tolerance $\epsilon_2 < \epsilon_1$
(4) Update the color models

Fig. 8. Final optimization algorithm.

For tracking, 160 points were chosen on the model's surface: 60 on the torso, 20 on each limb and on the head. More surface points were chosen on the torso because of its size and the fact that its 3D reconstruction is the least accurate one of all body parts. For the edge term, 40 points were chosen on the model contours: 10 on each limb, none on the torso and the head, giving a total of 120 points considering the contours of three camera views. Experiments showed that using edges for the torso and the head is not beneficial. Since the torso and the head yield quite salient blobs in the reconstruction, matching their surface against the reconstruction is more robust than using edges. Indeed, edges rather confuse the tracker due to the influence of the arms and legs on the torso's silhouette.

The tolerance ϵ_1 (Eq. (20)) for the full optimization was set to 2° for the angles and 2 cm for the model's displacement. For the refinement of the arms, legs and head the required accuracy was set to $\epsilon_2 = \frac{1}{2}\epsilon_1$. An intuitive way to initialize the step sizes for SMD is to observe the value of the gradient in the first few iterations to estimate the order of magnitude one wants to apply to each dimension. In our case, the initial step sizes were set as *2500* for all pitch and yaw angles. The roll angles were given higher values of *5000* since the objective function tends to be rather flat for these dimensions. The importance of the knees and elbows has been taken into account with a high initial step-size of *10,000*. The $\boldsymbol{\mu}$ vector also depends on the application, i.e. the gradient of the objective function. We found values of one-tenth of the initial step sizes working best. Indeed, increasing μ too much can cause the instability of the gradient descent for the step sizes and therefore of the system. λ regulates the time scale over which long-term dependencies are taken into account ($0 \leqslant \lambda \leqslant 1$). For our experiments a value of *0.3* proved to be best.

7.1. Stochastic vs. deterministic sampling

Shuffling the sample points on each iteration lets SMD escape from local minima more easily. A comparison between SMD and a deterministic version thereof (i.e. SMD without shuffling the points on each iteration) demonstrates this property. We have chosen a sequence with strong occlusions of the lower arms, as they are rotated around each other like the pedals of a bicycle, see Fig. 11. Then, the right arm of the model was displaced and both the stochastic and the deterministic version of SMD were used to bring it back into its correct position.

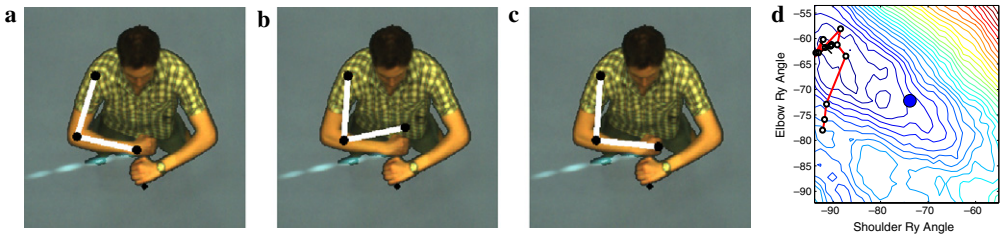


Fig. 9. A stochastic and a deterministic version of SMD were used to optimize the position of the arm. Image (a) shows the optimal placement of the arm, image (b) the starting position for the experiment. The optimization result for the deterministic version of SMD is shown in image (c). The arm is still out of place as SMD got stuck in a local minimum after 12 iterations. Plot (d) shows the iterations and the objective function where a local minimum at the upper left corner is seen to trap the arm pose in its basin of attraction. The optimum is the indicated point in the center.

For the experiment, the y-rotation angles of the right shoulder and the right elbow were moved from their optimal position $-73.9°$ and $-72.3°$ (Fig. 9a) to the position $-92.2°$ and $-78.0°$ (Fig. 9b)) resp. For tracking, 10 points on the arm surface and 10 points on each projected contour were chosen.

The deterministic version of SMD converges after 12 iterations to orientations -91.6 and -61.7, as shown in Fig. 9c. The plot (d) shows the objective function for a hyperplane through the parameter space for the two parameters to be optimized. The iterations are drawn as line segments. Obviously, the deterministic version of SMD was trapped in a local minimum and thus converged at a wrong position.

The same experiment was then repeated with the original SMD algorithm, i.e. with point shuffling. SMD converged after 15 iterations to position $-78.9°$ and $-70.7°$, obviously closer to the optimum than the deterministic version. Fig. 10 shows the end position and the objective function for iterations 3, 8, 11, and 15. The objective function changes at each iteration. The spurious local minima tend to move around, while the true optimum stays put. This gives SMD the chance to hop out of a local minimum at each iteration.

7.2. Single vs. multiple cues

In [19] we showed that tracking of complex motions is possible even if only the model's surface is matched to the reconstructed one. The 3D color segmentation was the key element to successfully disambiguate difficult situations when limbs come close to each other. However, color information must be available to benefit from the 3D color segmentation. Otherwise, the tracking can fail and the color models can be destroyed as they are updated with the wrong colors.

An example of such a situation is given in Fig. 11 where the person brings his arms close to the body and creates occlusions. The uppermost row shows the noisy reconstruction of three frames of the sequence, with many artifacts. The gap between the arms and the torso is closed and the color of the shirt is rather similar to skin color. Tracking with the surface alignment only is problematic in this case as shown in the second row: the left arm gets stuck to the body for some frames. The bulky reconstruction gives no clue about the exact placement of the arms.

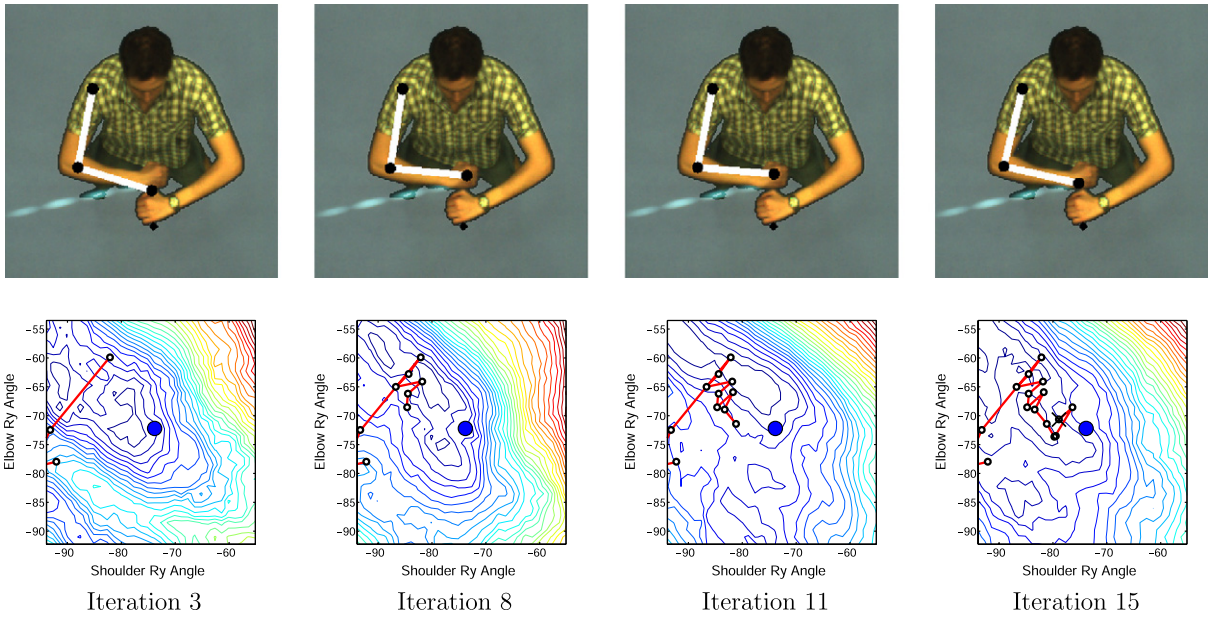


Fig. 10. Stochastic sampling overcomes the local minima as shuffling the points on each iteration lets the spurious local minima move around. The resulting solution after 15 iterations (at black cross) is closer to the optimum (centered circle) then for the deterministic version.

The third row shows the results obtained with the same configuration but with the edge alignment enabled. The tracker is successful during tracking the whole sequence and the accuracy is clearly better. Yet, the placement of the arms is not perfect for some frames of the sequence. This is caused by the slight twist of the torso which the tracker can not pick up from the bulky reconstruction.

7.3. Comparison to Levenberg–Marquardt

In Section 7.1 we have shown that shuffling the points at each iteration lets SMD escape from local minima. Moreover, using only a subset of the points, the tracking gains significantly in speed. To prove our claims we want to compare SMD to the widely used Levenberg–Marquardt (LM) method. The speed and accuracy of both methods were compared for a challenging test sequence.

The test sequence (115 frames) shows a person performing full articulation. Some frames are shown in Fig. 12. First, the person puts his hands on the hips and then he kicks with both legs while slightly turning and keeping the arms in a defensive manner close to the upper body. The tracker must be able to keep the arms separated from the torso and should survive the fast motions. Furthermore, the person leans backwards and turns to the left while the extremities move fast. The reconstruction of the sequence was computed with a voxel space resolution of 64^3.

Reference poses for the test sequence were generated using a semi-automatic procedure. The sequence was tracked with a gradient descent and considering all points on the model's surface and contours. Manual intervention was used to correct the pose when a misplacement could be visually noticed. We consider these reference poses to be groundtruth to compare against.

For our comparison with LM optimization, the same number of points were used. Furthermore, while tracking with LM, the set of points is shuffled once for each new frame before optimization is performed.

SMD was successful in tracking the whole sequence as can be seen for the five selected frames in Fig. 12b). The average number of iterations was computed as the mean of all six optimizations (one for the full optimization and

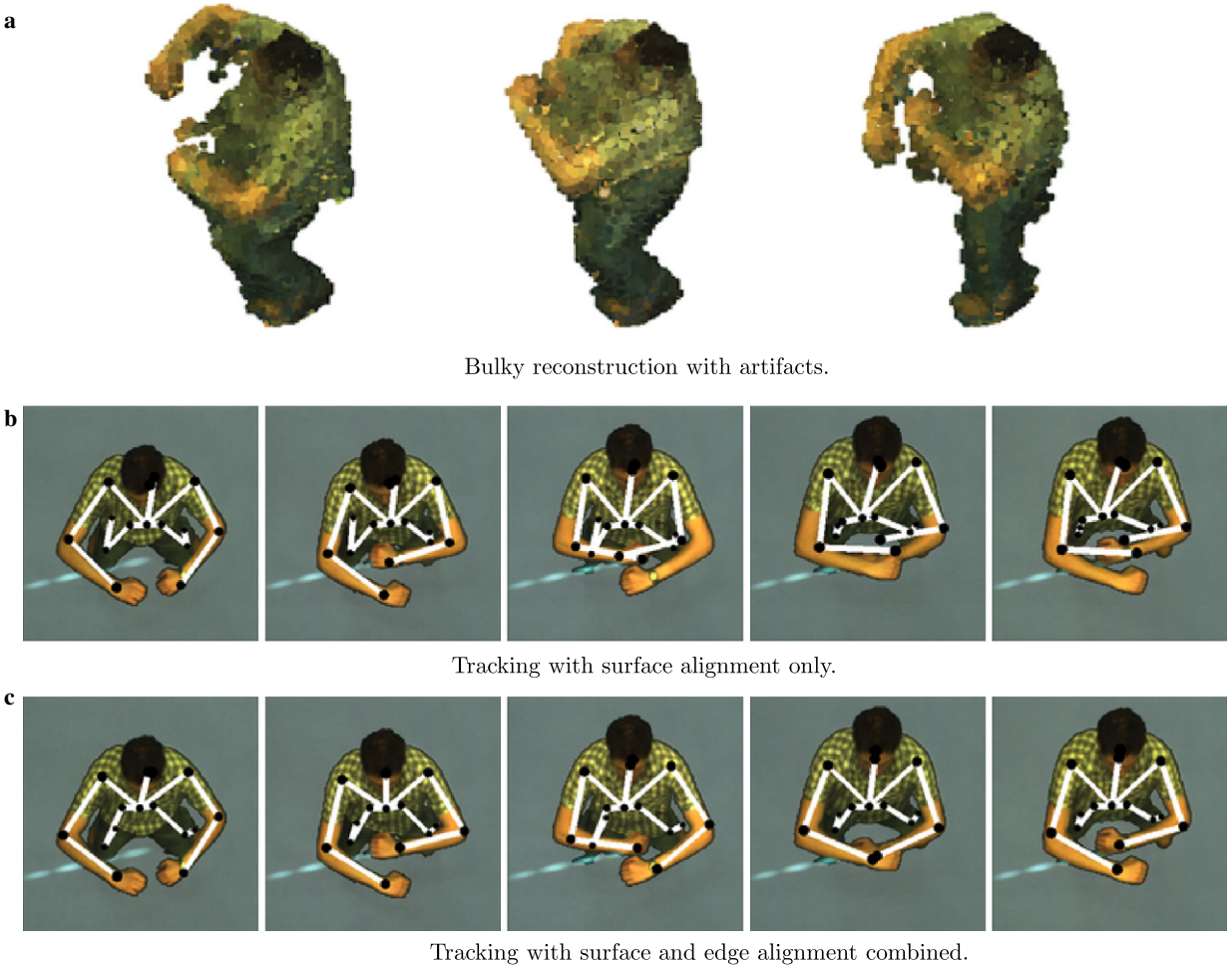


Fig. 11. Tracking a sequence with inaccurate reconstruction (a) and using only surface alignment is problematic if limbs come close (b). When incorporating edges the tracker performs well and more accurate over the full sequence (c).

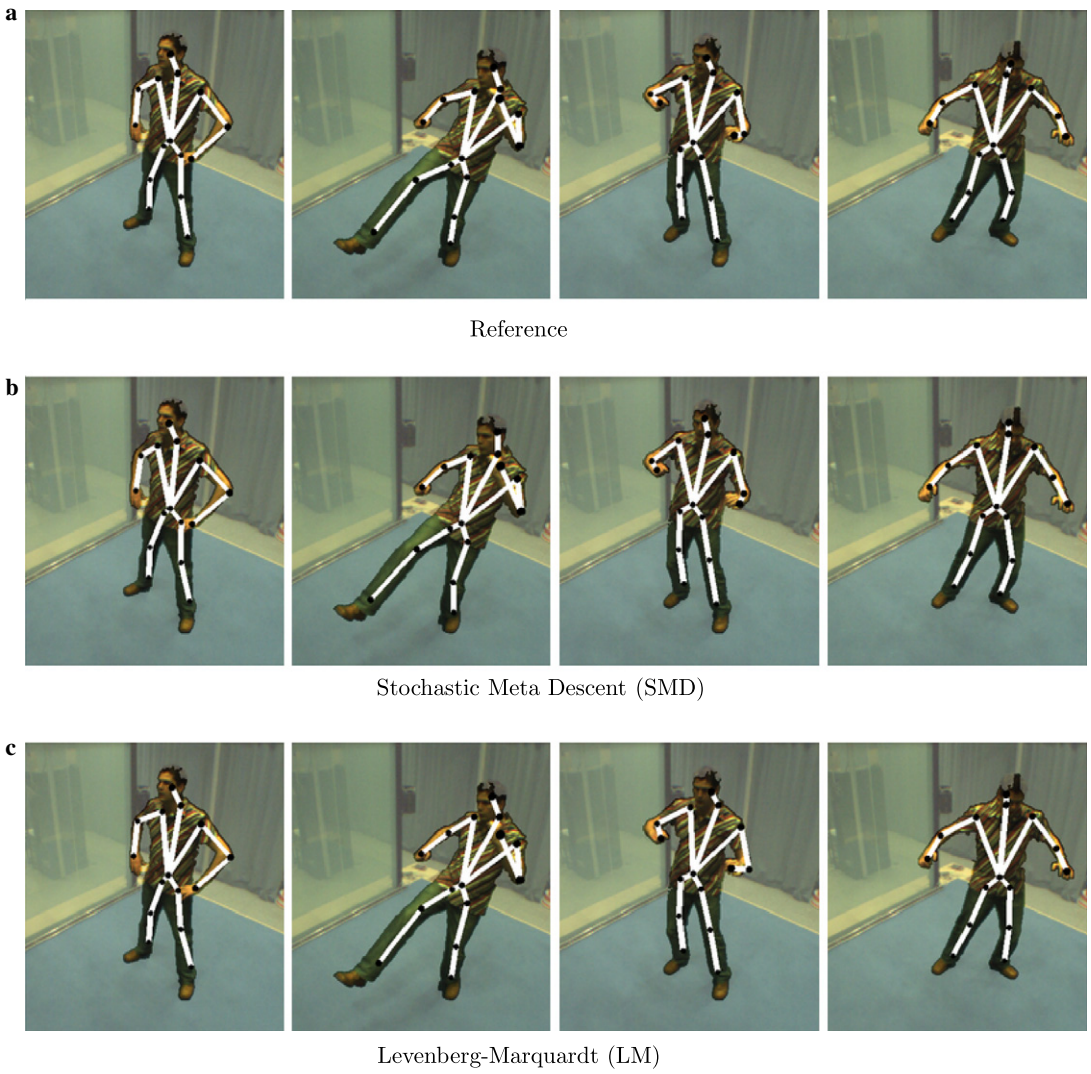


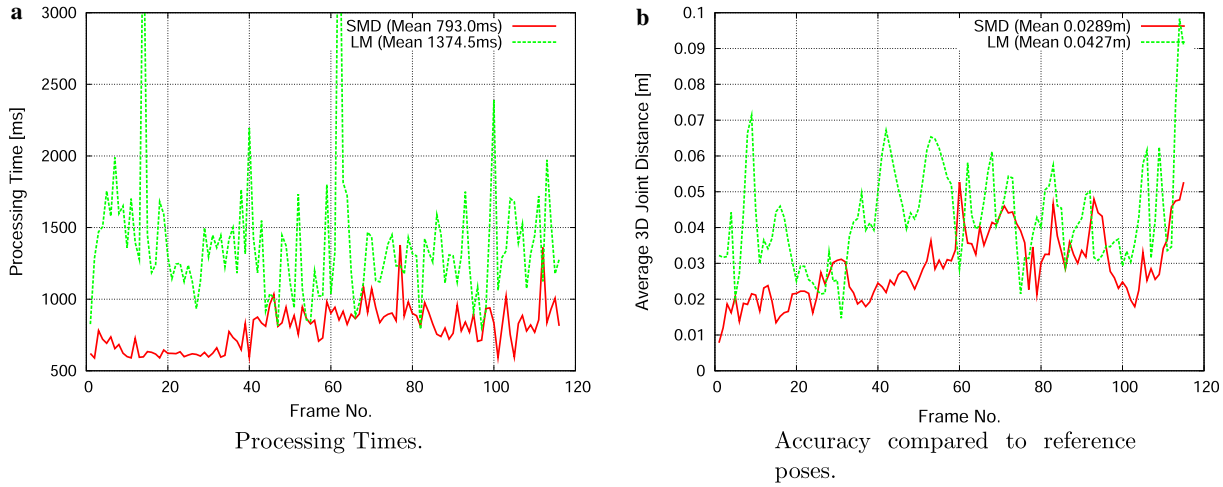
Fig. 12. Comparison between Levenberg–Marquardt (LM) and stochastic meta descent (SMD) optimization. The first row shows the reference poses for five keyframes (15, 55, 75, 115) and the second row the poses obtained by tracking with SMD. LM could only succeed if five times as many points were used for tracking than for SMD. The results for LM are shown in the last row.

five for the refinement of the head, arms and legs) over all frames. On average SMD needed 6 iterations to converge where the first optimization of all DoFs usually needed slightly more iterations. The processing time per frame was obtained by adding the computation time of the full optimization, the slowest refinement time, 45 ms for the network overhead, and 15.02 ms for the volumetric reconstruction. SMD was able to track the sequence with an average processing time of 793.0 ms per frame what corresponds to a framerate of 1.3 fps. The processing times for the individual frames are shown in Fig. 13a).

LM was not able to track the sequence with the same number of points. After 25 frames, LM misplaced the torso and consequently lost track of the left arm. We had to increase the number of points to at least five times more than SMD needed to achieve robust tracking. Therefore, the experiment was repeated with five times more points for tracking. This time, LM was successful in tracking the whole sequence as can be seen for the five selected frames in Fig. 12c). On average, LM needed 9 iterations and 1374.5 ms per frame, corresponding to 0.73 fps. The processing times for the individual frames can also be seen in Fig. 13a). For some frames LM needed far more iterations than average to converge what is undesirable for tracking. In contrast, SMD's processing times were more constant.

The experiment has demonstrated that SMD is able to track a complex sequence with fewer points and is faster as a consequence. However, the tracking accuracy should also be considered. We will do this by comparing the resulting poses of both methods to our set of reference poses.

An obvious method to compare two poses is to consider the Euclidean distance between their pose vectors $\mathbf{p}$. However, this measurement does not necessarily correspond to the perceptually difference. As a more appropriate measure we propose the mean distance between the groundtruth 3D joint positions and the computed ones. Fig. 13b) shows the computed distance errors for both optimizers over the test sequence. The distance error of SMD is 2.89 cm, the one



Processing Times. Accuracy compared to reference poses.

Fig. 13. Plot (a) shows the processing times for SMD and LM over the test sequence. SMD is obviously faster as fewer points were sufficient for tracking. Plot (b) shows the mean distance of the expected 3D joint positions and the computed ones. The distance error of SMD is 2.89 cm, the one achieved by LM is 4.27 cm. Outliers often result from bad localization of the torso.

achieved by LM is 4.27 cm. Strong outliers are mostly caused by bad localization of the torso. The decrease of accuracy starting at frame 40 results from the left turn of the person. During this left turn the orientation of the person is some times not optimal for our camera alignment what becomes apparent in a bulky reconstruction. Consequently, optimal alignment of the torso is difficult.

The tracking result of SMD for additional frames can be seen in Fig. 15. After the test sequence the person performs a full 360° turn and brings the arms close to the body. Again, during the turn the placement of the torso is sometimes not optimal as the reconstruction is quite bulky and inaccurate. Consequently, the arms can not be brought into perfect position due to their fixed length. However, SMD succeeds in tracking this complex sequence.

7.4. Fast motions

Fast motions are critical for tracking as they require the optimization to find the optimum far away from the starting point. This involves the risk of divergence or even of convergence to the wrong position. For instance, if an arm moves fast away from the torso, the 'optimal' solution found by the optimization could be to put the arm on the torso, although there would be a better solution but further away.

The sequence presented here was used to test the robustness of our tracker during fast movements and is 375 frames long. Our system was successful in tracking this sequence as can be seen in Fig. 16. The average processing time was 788.4 ms or 1.27 fps. However, the achieved accuracy of the pose estimates is clearly worse compared to slower motions, especially for the torso parameters.

The sequence is divided into three parts. During the first part (frames 1–95), the person dances and swings the arms around. In the second part of the sequence (frames 96–175) the person walks fast on the spot. The walking motion can be nicely seen at the pitch angles of the hip joints, as illustrated in Fig. 14 a), Fig. 16. In the last part of the sequence, the person adopts different extreme poses with fast transitions in between. In doing so, the joint angles of the shoulders undergo large changes as illustrated in Fig. 14b) for

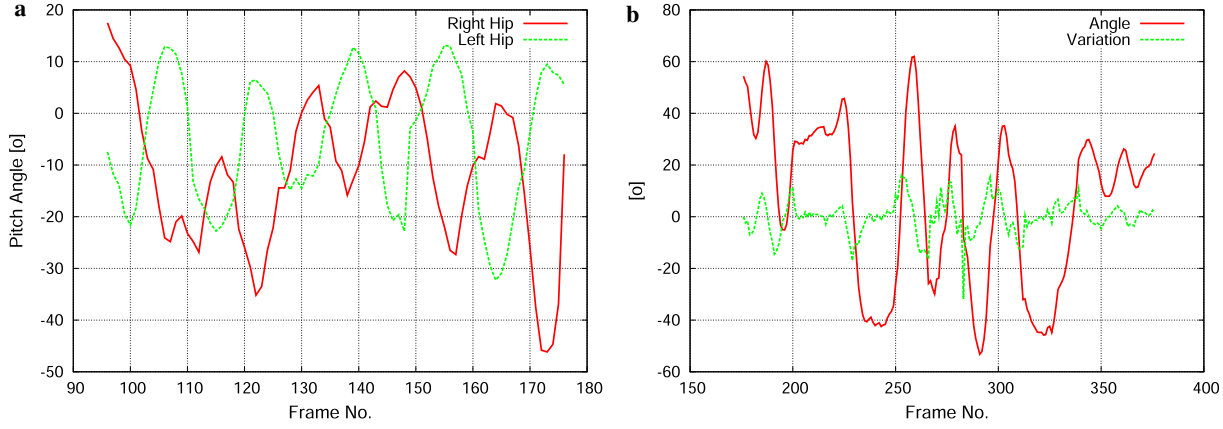


Fig. 14. Plot (a) shows the pitch angles of the hip joints during the walking motion. Plot (b) shows the pitch angle of the right shoulder joint and its variation during fast motions. The parameter changes up to 31° between consecutive frames.

the pitch angle of the right shoulder joint. During these fast motions the parameter changes up to 31° between consecutive frames. However, the average change is 4.1° during the whole sequence.

7.5. Presence of an object

Next we test the performace of our method in presence of an object. The captured sequence is 200 frames long and shows a person playing with a cube. The cube is passed from one hand to the other, thrown against a wall, kicked, and finally pickup up again. Again, fast motions of the limbs can be observed. The resolution of the voxel space was set to 128^3 to illustrate the effect of such higher resolution on the tracking.

The results are shown in Fig. 17. SMD had no problem to track this sequence. The average processing time was 1131.0 ms or 0.8 fps including 70 ms for the computation of the reconstruction. The processing time for tracking increased as an effect of the higher voxel space resolution and, as a consequence, the larger set of voxels which had to be considered for the nearest-neighbor search.

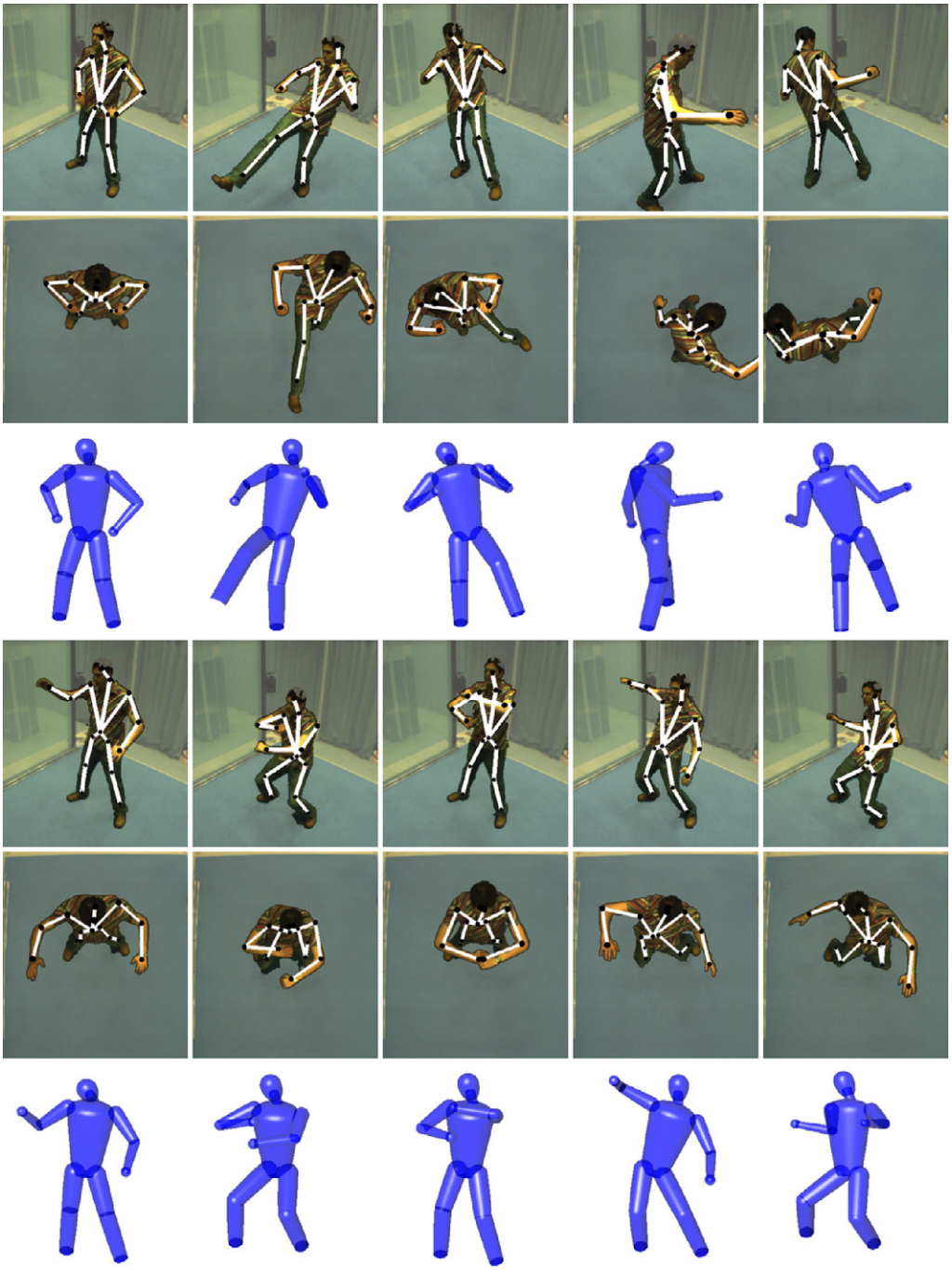


Fig. 15. Tracking result of a sequence where the person performs full articulation and turns around its own axis.

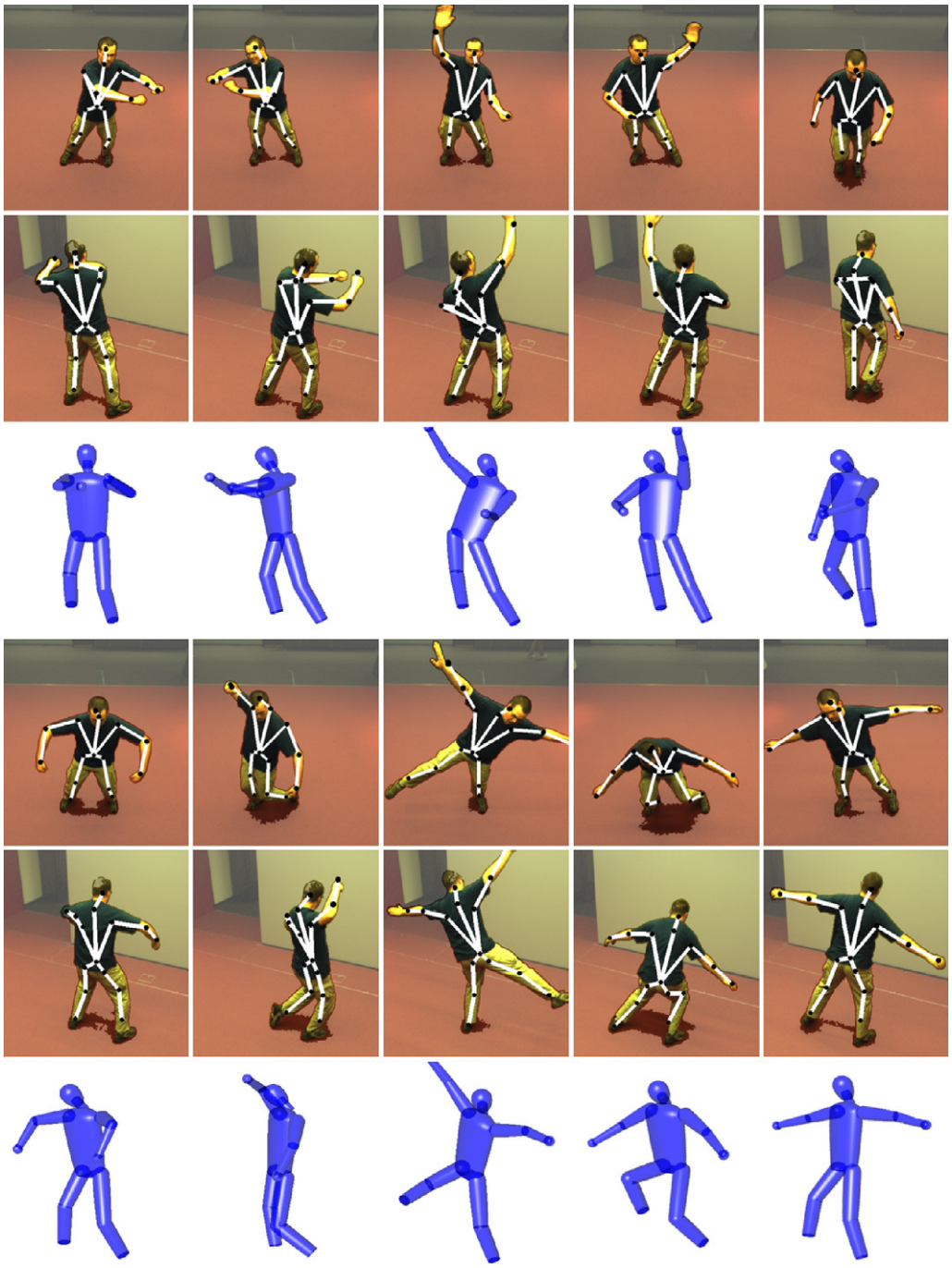


Fig. 16. Tracking result of a sequence where the person first performs fast dancing motions and then adopts extreme poses with fast transitions in between.

8. Conclusions

In this paper, we presented an algorithm capable of tracking a person's full body pose during complex motions without any markers. We follow a model-based approach by fitting a body model to the images of several synchronized video streams. For model-fitting, multiple image cues are combined into an objective function which yields robust tracking.

We propose an efficient method to compute the volumetric reconstruction of the person. We have shown that voxel-based procedures can achieve real-time reconstruction. Instead of projecting each voxel onto the image plane of each cameras we keep lookup-tables for the inverse mapping: they hold a list for each pixel containing all voxels which are projected on it. This brings two advantages: (1) the lookup-tables can be computed off-line and thus no projections have to be done during runtime and (2) the reconstruction can be updated for consecutive frames instead of being extracted from scratch.

The volumetric reconstruction is the strongest cue in our tracking framework. However, in difficult situation such as

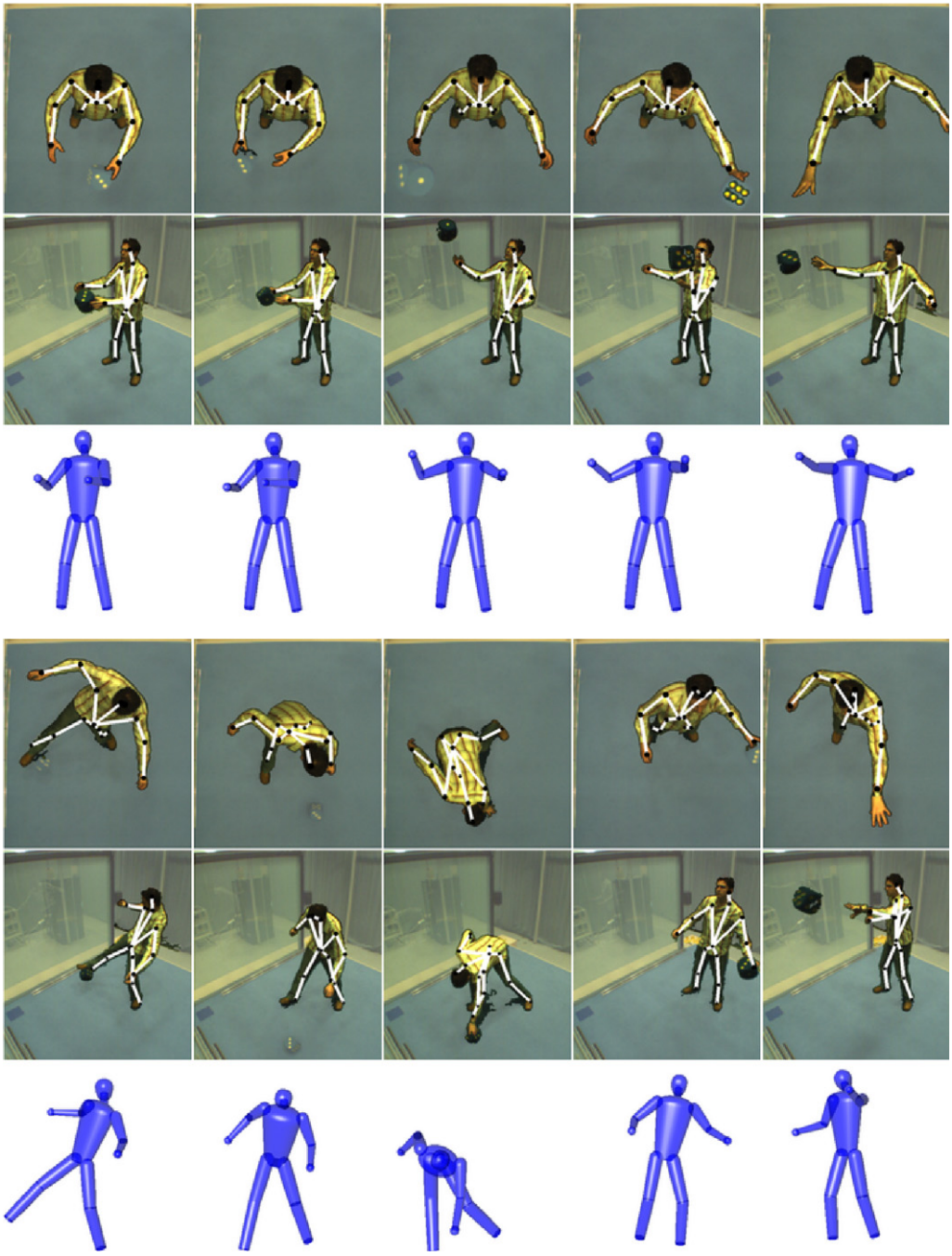


Fig. 17. Tracking result of a sequence where the person plays with a cube.

when limbs come close, tracking becomes ambiguous. Therefore we combine multiple cues. The image edges offer good localization for the model and help to disambiguate the solutions. We use color edges as they produce less clutter because of shadows or wrinkles on the clothes. Moreover, we perform a 3D color segmentation of the reconstruction. This segmentation lowers the computational costs for the surface matching by discarding unlikely voxels and further strengthens the tracking by separating limbs.

For optimization we propose to use SMD. Its stochastic sampling, i.e. shuffling the points on each iteration, lets SMD overcome local minima and increases tracking speed at the same time. Spurious minima tend to move around while the global optimum stays fixed.

We illustrated the performance of our method by several challenging sequences where a person wears different, casual clothes. Five cameras were sufficient for tracking the complex motions. The multi-cue approach proved effective in providing the required robustness. The small set of points needed for optimization with SMD and our efficient reconstruction algorithm made it possible to track at about 1 frame per second. The comparison of SMD to the commonly used Levenberg–Marquardt proved our claim that

the use of stochastic optimization is beneficial in terms of speed and robustness.

The videos of the experiments presented here can be downloaded from our webpage *http://www.vision.ee.ethz.ch/~rkehl/Videos/*

Acknowledgments

This work is carried out in the context of the blue-c-II project, funded by ETH Grant No. 0-21020-04 as an internal poly-project and by the Sixth Framework Programme of the European Commission: EU Project FP6 511092 (CyberWalk).

References

[1] A.H. Barr, Superquadrics and angle-preserving transformations, IEEE Comput Graph. Appl. 1 (1981) 11–22.
[2] J.E. Bresenham, Algorithm for computer control of a digital plotter, in: IBM Systems Journal, vol. 4(1), 1965, pp. 25–30.
[3] C. Bregler, J. Malik, Tracking people with twists and exponential maps, in: Proc. of CVPR, 1998, pp. 8–15.
[4] J. Carranza, C. Theobalt, M. Magnor, H.P. Seidel, Free-viewpoint video of human actors, in: Proc. of ACM SIGGRAPH, 2003, pp. 569–577.
[5] K.M. Cheung, T. Kanade, J.-Y. Bouguet, M. Holler, A real time system for robust 3D voxel reconstruction of human motions, in: Proc. of CVPR, vol. 2, 2000, pp. 714–720.
[6] K.M. Cheung, S. Baker, T. Kanade, Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture, in: Proc. of CVPR, vol. 1, 2003, pp. 77–84.
[7] D. Demirdjian, Enforcing Constraints for Human Body Tracking, in: Proc. of CVPR, vol. 9, 2003, pp. 102–109.
[8] Q. Delamarre, O.D. Faugeras, 3D articulated models and multi-view tracking with silhouettes, in: Proc. of ICCV, vol. 2, 1999, pp. 716–721.
[9] Q. Delamarre, O.D. Faugeras, 3D articulated models and multiview tracking with physical forces, in: CVIU Special Issue on Modeling People, vol. 81(3), 2003, pp. 328–357.
[10] J. Deutscher, A. Blake, I. Reid, Articulated body motion capture by annealed particle filtering, in: Proc. of CVPR, 2000, pp. 126–133.
[11] J. Deutscher, A. Davison, I. Reid, Automatic partitioning of high dimensional search spaces associated with articulated body motion capture, in: Proc. of CVPR, vol. 2, 2001, pp. 669–676.
[12] H. Dreyfuss, The Measure of Man: Human Factors in Design, Whitney Library of Design, New York, 1959–1967.
[13] D.M. Gavrila, L.S. Davis, 3D model-based tracking of humans in action: a multi-view approach, in: Proc. of CVPR, 1996, pp. 73–80.
[14] A. Griesser, De S. Roeck, A. Neubeck, L. Van Gool, GPU-based foreground-background segmentation using an extended colinearity criterion, in: Proc. of Vision, Modeling, and Visualization (VMV), 2005, pp. 319–326.
[15] R. Hartley, A. Zisserman, Multiple View Geometry in Computer Vision, Cambridge University Press, 2000.
[16] J. Hasenfratz, M. Lapierre, J. Gascuel, E. Boyer, Real-time capture, reconstruction and insertion into virtual world of human actors, Vis. Video Graph. Conf. (2003) 49–56.
[17] I.A. Kakadiaris, D. Metaxas, 3D human body model acquisition from multiple views, in: Proc. of ICCV 1995, pp. 618–662.
[18] I.A. Kakadiaris, D. Metaxas, Model-based estimation of 3D human motion with occlusion based on active multi-viewpoint selection, in: Proc. of CVPR, 1996, pp. 81–87.
[19] R. Kehl, M. Bray, L. Van Gool, Full body tracking from multiple views using stochastic sampling, in: Proc. of CVPR, 2005, pp. 129–136.
[20] D. Metaxas, D. Terzopoulos, Shape and nonrigid motion estimation through physics-based synthesis, in: TPAMI, vol. 15(6), 1993, pp. 580–591.
[21] I. Mikic, M. Trivedi, E. Hunter, P. Cosman, Articulated body posture estimation from multi-camera voxel data, in: Proc. of CVPR, Vol. 1, 2001, pp. 455–460.
[22] I. Mikic, M. Trivedi, E. Hunter, P. Cosman, Human body model acquisition and tracking using voxel data, in: IJCV, vol. 53(3), 2003, pp. 199–223.
[23] J.R. Mitchelson, A. Hilton, Simultaneous pose estimation of multiple people using multiple-view cues with hierarchical sampling, in: Proc. of BMVC, 2003.
[24] R. Mester, T. Aach, D. Dümbgen, Illumination-invariant change detection using a statistical colinearity criterion, in: Proc. of DAGM, vol. 23, 2001, pp. 170–177.
[25] R. Plaenkers, P. Fua, Model-based silhouette extraction for accurate people tracking, in: Proc. of ECCV, vol. 2, 2002, pp. 325-331.
[26] T.J. Roberts, S.J. McKenna, I.W. Ricketts, Online appearance learning for 3D articulated human tracking, in: Proc. of ICPR, vol. 1, 2002, pp. 425–428.
[27] P. Sand, L. McMillan, J. Popovic, Continuous capture of skin deformation, ACM Trans. Graph. 22 (3) (2003) 578–586.
[28] N.N. Schraudolph, Local gain adaptation in stochastic gradient descent, in: Proc. Intl. Conf. Artificial Neural Networks, 1999, pp. 569–574.
[29] H. Sidenbladh, M.J. Black, D.J. Fleet, Stochastic tracking of 3D human figures using 2D image motion, in: Proc. ECCV, 2000, pp. 702–718.
[30] C. Sminchisescu, B. Triggs, Covariance scaled sampling for monocular 3D body tracking, in: Proc. of CVPR, 2001, pp. 447–454.
[31] C. Sminchisescu, Consistency and coupling in human model likelihoods, in: Proc. of Face and Gesture Recognition, 2002, pp. 27–32.
[32] T. Svoboda, D. Martinec, T. Pajdla, A convenient multicamera self-calibration for virtual environments, In Presence: Teleoperators Virtual Environ. 14 (4) (2005) 407–422.
[33] R. Szeliski, Rapid octree construction from image sequences, Comput. Vis. Graph. Image Process. 58 (1) (1993) 23–32.
[34] C. Theobalt, J. Carranza, M. Magnor, H.P. Seidel, Enhancing silhouette-based human motion capture with 3D motion fields, in: Proc. of Pacific Graphics, 2003, pp. 185–193.
[35] P. Trahanias, A.N. Venetsanopoulos, Color edge detection using vector statistics, IEEE Trans. Image Process. 2 (1993) 259–264.
[36] S. Wachter, H.H. Nagel, Tracking persons in monocular image sequences, Comput. Vis. Image Understand. 74 (3) (1999) 174–192.
[37] S. Wesolkowski, E. Jernigan, Color edge detection in RGB using jointly euclidean distance and vector angle, in: Proc. of the IAPR Vision Interface Conference, 1999, pp. 9–16.
[38] Y. Yang, Colour edge detection and segmentation using vector analysis, Master's thesis, Electrical and Computer Engineering, University of Toronto, Toronto, Ont., Canada, 1995.