Robotics and Autonomous Systems 75 (2016) 539–549

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Semantic parametric body shape estimation from noisy depth sequences

Alexandru Eugen Ichim [a], Federico Tombari [b,c,*]

[a] École Polytechnique Fédérale de Lausanne, Switzerland
[b] University of Bologna, Italy
[c] TU Munich, Germany

HIGHLIGHTS

- A framework for tracking and modeling of human bodies from sequences of depth maps.
- Modular and extensible energy cost optimization, with depth and prior constraints.
- Compact semantic tags associated to the estimated body shape using L1 relaxation.
- Relies on the tools and algorithms provided by the Point Cloud Library (PCL).
- 3 fps performance for continuous tracking and modeling on the CPU.

ARTICLE INFO

Article history:
Available online 6 November 2015

Keywords:
3D body modeling
3D body tracking
Depth data
Point Cloud Library

ABSTRACT

The paper proposes a complete framework for tracking and modeling articulated human bodies from sequences of range maps acquired from off-the-shelf depth cameras. In particular, we propose an original approach for fitting a pre-defined parametric shape model to depth data by exploiting the 3D body pose tracked through a sequence of range maps. To this goal, we make use of multiple types of constraints and cues embedded into a unique cost function, which is then efficiently minimized. Our framework is able to yield compact semantic tags associated to the estimated body shape by leveraging on semantic body modeling from MakeHuman and L1 relaxation, and relies on the tools and algorithms provided by the open source Point Cloud Library (PCL), representing a good integration of the functionalities available therein.

1. Introduction and related work

The task of 3D body modeling aims at automatically obtaining an accurate 3D model of a person's body. The possibility of having at disposal an accurate 3D model adapted to the body characteristics of a subject opens up new directions in a variety of applications, such as in the fields of entertainment (e.g. 3D avatar creation for videogaming and movie special effects), fitness (e.g., for automatic estimation of the body mass), apparel (e.g., for virtual changing room applications), interactive design, and security (people detection and identification).

The output of this task is generally represented by a parametric 3D body model, with the parameters estimated so that the model adapts to the specific characteristics of the subject being scanned. It is often the case that these parametric models are open sourced and available to the community so to favor interchange and standardization. While earlier parametric models [1] were based on simple Principal Component Analysis (PCA) of standard human poses (e.g., T/A poses), more recent approaches also model minute body deformations such as muscle bulging under complex poses, e.g., the SCAPE models [2,3]. Another possibility of sourcing parametric body models is from semantic models, i.e., models built by artists, where each body shape modifier has an associated semantic tag, such as it is the case of MakeHuman [4].

Accurately estimating the 3D body model traditionally requires dedicated and expensive hardware to acquire high resolution scans of the body, generally by means of 3D laser scanners or high frame-rate structured light sensors. In addition, this procedure is characterized by high processing time due to the re-positioning of the scanner from different view points, the acquisition and the joint 3D registration of the different scans. To overcome

* Corresponding author at: University of Bologna, Italy.
E-mail address: federico.tombari@unibo.it (F. Tombari).

http://dx.doi.org/10.1016/j.robot.2015.09.029
0921-8890/© 2015 Elsevier B.V. All rights reserved.

such limitations, the work of [5] proposed to fit a 3D parametric model to a frame acquired by means of a monocular RGB camera. Although not fully automatic due to the need of user interaction as well as limited in the modeling accuracy due to the 2D to 3D fitting, this work introduced the concept of using low-cost hardware for the task of 3D body modeling.

Successively, thanks to the popularity of consumer depth cameras originated by the development of the Microsoft Kinect, other works [6–9] have tackled 3D body modeling by means of the noisy range data acquired from such low-cost 3D sensors. Initially, [6] proposed to fit each parametric 3D model obtained from SCAPE [2] on a certain number of range depth maps (e.g. 4) by optimizing an objective cost function relying on 3D data fitting as well as silhouette fitting. The main limitations of such a method are represented by the constraints imposed by the system, in the form of a specific pose (*T-pose*) that the subject has to assume throughout the sequence, and by the overall efficiency (more than one hour is reported to process one subject). Successively, in [8], simplified SCAPE shape models are estimated from two depth maps of the subject (one frontal, one from the back) in real time by optimizing a cost function composed of two terms, respectively taking into account point-to-point and point-to-plane fitting. Analogously to [6], this method carries out the modeling by relying on a small number of slightly overlapping frames, hence might suffer from the presence of noise in the data.

Differently, non-parametric shape modeling approaches have been also proposed. This is the case of [7], where a moving voxel grid is used for each body part to integrate together surface measurements obtained from a depth map sequence within a Truncated Signed Distance Function (TSDF) representation, thus allowing to build volumetric models for both the background and each piecewise body part. Due to the TSDF fusion, the output is not a parametric body model, but a piecewise smooth 3D mesh reconstruction of the body. Another non-parametric approach is the one proposed in [9], where real-time pose and shape estimation is obtained via a probabilistic approach based on a Gaussian Mixture Model (GMM). Also in this case, the input is represented by a sequence of RGB-D frames. A purely point-based technique is proposed by [10] for the people re-identification task; the authors use the Microsoft SDK to track and segment the body, and then the points are accumulated by transforming each limb to the standard A-pose.

In this work, we propose a framework aimed at efficient 3D parametric body modeling from noisy depth sequences acquired with consumer depth cameras. Conversely to [6,8], one main contribution of this work is to leverage on the temporal cue by explicitly tracking the 3D subject and estimating its 3D pose through a sequence of frames. This allows us to integrate the noisy body shape of the subject over many temporally correlated frames, effectively averaging out noise. The modeling procedure is carried out by minimization of an energy cost which includes, as a second contribution of our approach, additional set of cues with respect to those used in previous works, based on silhouette and 3D surface fitting, as well as skeleton similarity, PCA and smoothness. We show that the combinations of these terms induce a more robust estimation of the body model. Finally, and differently to [6,8], we propose to use MakeHuman models [4] due to their better integration with semantic information associated to each body part. In conjunction with this, a third contribution is a specific L1 minimization of the energy cost term associated with our modeling scheme, so as to induce sparsity in the semantic tags and automatically yield a compact semantic description of each acquired body.

In Sections 2 and 3 we illustrate the entire proposed pipeline, which tracks the body motion of the subject and estimates the pose of its body joints over time, then, from the 3D estimated body pose at each frame, it refines the parameters of a MakeHuman body model by cost function optimization. A graphical overview of the proposed pipeline is shown in Fig. 1. Our framework relies on open-source computer vision and full body modeling libraries such as the Point Cloud Library (PCL) and MakeHuman, and it is easily customizable for different tasks requiring different precision and performance due to the modularity of its nature. In our initial implementation it is able to process frames at a speed of 3 fps and does not require specific constraints on the pose of the subject. To demonstrate the effectiveness of our approach, in Section 4 we show some qualitative examples of body models estimated and tracked from real data acquired from consumer depth cameras, as well as measured accuracy of the estimated body model with respect to specific body parts. We also demonstrate the usefulness of compact semantic body tags associated to our estimated body models.

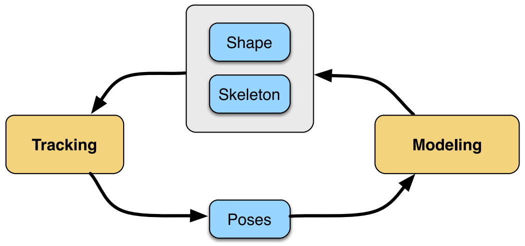


Fig. 1. Overview of the proposed tracking and modeling pipeline.

2. Proposed methodology

2.1. Data representation

In our system, the articulated human bodies are represented as quad and/or triangle meshes. The bodies can be articulated via the underlying skeleton. Skeletons are composed of multiple joints disposed in a tree hierarchy (see Fig. 2 for an example), based on which local node transformations are propagated: $\boldsymbol{T}^j_{abs} = \boldsymbol{T}^{parent(j)}_{abs}\boldsymbol{T}^j_{local}$, where $\boldsymbol{T}^j_{abs}$ is the world transformation of joint j, and $\boldsymbol{T}^j_{local}$ is its local transformation with respect to its parent node in the skeleton tree. Each joint influences a number of mesh vertices in its vicinity, as defined by the linear blend skinning model: $\boldsymbol{v}^{pose}_i = \sum_j w^j_i \boldsymbol{T}^{pose}_j * (\boldsymbol{T}^{rest}_j)^{-1} * \boldsymbol{v}^{rest}_i$, where each joint j in the skeleton has $\boldsymbol{T}^{rest}_j$ as the transformation corresponding to the rest pose (A or T-pose) and $\boldsymbol{T}^{pose}_j$ the transformation of the joint in the posed skeleton configuration, and w^j_i is the blend skinning weight of joint j over mesh vertex $\boldsymbol{v}_i$. The joints are modeled by their rest transformation (expressed using rotation matrix $\boldsymbol{R}^{rest}_j$ and translation vector $\boldsymbol{t}^{rest}_j$) and the pose rotation parametrized using Euler angles $\boldsymbol{\beta}$: $\boldsymbol{T}^{pose}_j = \boldsymbol{R}^{rest}_j * \boldsymbol{R}^x(\beta^x_j) * \boldsymbol{R}^y(\beta^y_j) * \boldsymbol{R}^z(\beta^z_j) + \boldsymbol{t}^{rest}_j$.

The skeleton model deforms the mesh based on the pose of the body, but does not take into account the deformations that define the identity of a person. To this end, we employ a global linear deformation model in which vertices $\boldsymbol{v}^{rest}_i$ are expressed as linear combinations $\boldsymbol{s}$ of bases stacked as columns into matrix $\boldsymbol{B}$: $\boldsymbol{v}^{rest}_i = \boldsymbol{m}_i + \boldsymbol{B}_i\boldsymbol{s}$. Previous work such as [1–3] uses statistical models derived from a set of registered scans of people. The framework we propose allows for such models to be used (they use the same linear system), but in our implementation we employed blendshapes exported from the popular human body modeling software MakeHuman [4]. These blendshapes correspond to the sliders in the MakeHuman application, that is used by numerous artists and game developers to generate realistic character assets. As a result, our model is based on a set of non-orthogonal bases

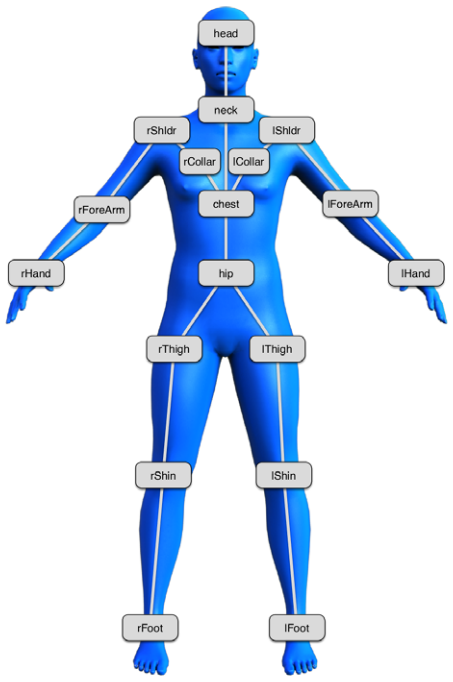


Fig. 2. MakeHuman parametric body model with an overlaid example skeleton.

(blendshapes) that are linearly combined in order to obtain novel shapes. The advantage of using such a technique as opposed to statistical models is that the fitting weights $\boldsymbol{s}$ represent a certain comprehensible body characteristic along each dimension (e.g.: *fat/slim middle*, *more/less muscular* etc.). This enables applications such as body shape retargeting, where the body parameters computed with our system could be used by artists to model bodies in semantically equivalent spaces (e.g., use the human body blendshape weights to generate a semantically equivalent cartoonish character, animal or monster) (see Section 2.7 for more details).

We formulate the tracking and modeling problem as a global energy minimization, which aims to estimate the pose of the skeleton $\boldsymbol{\beta}_i$ at each frame i, as well as the global shape of the body, encoded by $\boldsymbol{s}$: $\text{argmin}_{\{\boldsymbol{\beta}_i\},\boldsymbol{s}} E_{total}$. Fig. 3 shows the different energies that we propose to use in our framework. In the following subsections we will explain the formulation of each energy functional, as well as offer an intuition on its contribution to solving the global problem.

2.2. Feature constraints

In order to start with a good initial alignment and anchor the tracking, we use a soft energy that keeps the global translations of each joint close to the tracked sparse set of body landmarks. Several off-the-shelf solutions are available for tracking body landmarks over depth sequences. In our experiments, we have used the Primesense NiTE body tracker (currently not available due to the acquisition of the company). Alternatives are represented by the People Tracking module in PCL, as well as the Microsoft Kinect SDK.[1] All these trackers process, as input, a sequence of depth maps as those provided by a consumer depth camera, and output, at each frame, a set of tracked points representing the skeletal joints of the human body appearing in the sequence. As such, any of these methods could be used within our framework for the goal of tracking 3D body landmarks.

[1] https://www.microsoft.com/en-us/kinectforwindows/.

The energy term modeling the alignment between each template body joint and the respective estimated body landmark via tracking is formulated as follows:

$$E_{features} = \sum_j \left\| \boldsymbol{t}_j^{pose} - \boldsymbol{\chi}_j \right\|_2^2 \qquad (1)$$

where $\boldsymbol{t}_j^{pose}$ are the world-space translations of each joint, and $\boldsymbol{\chi}_j$ are the corresponding tracked 3D features.

2.3. Point-to-plane constraints

The sampled scene surface obtained from the scanner is registered against the current estimate of the template body model. In order to align those two surfaces, the point-to-plane error metric is used:

$$E_{surface} = \sum_i \left\| \boldsymbol{n}_i^T (\boldsymbol{x}_i - \boldsymbol{v}_i) \right\|_2^2 \qquad (2)$$

where scan point $\boldsymbol{x}_i$ with its normal $\boldsymbol{n}_i$ is in correspondence with the template vertex $\boldsymbol{v}_i$. Mesh vertices $\boldsymbol{v}_i$ are expressed as functions of the skeleton pose and linear blend skinning, as explained in the previous subsection.

The PCL library offers multiple techniques for pre-processing the input data, obtaining and filtering pairs of corresponding points between the mesh and the depth map. The depth maps are represented as organized grids of 3D points, offering the possibility of using fast techniques such as integral images [11] to compute the normals of the depth maps efficiently. Furthermore, the input point cloud comes from a sensor that can be approximated via the pinhole camera model, the correspondences can be estimated in linear time by projecting the template vertices onto the depth map. Filtering is performed by discarding correspondences between points with incompatible distances and orientations [12]. The correspondences are computed at each outer iteration of the tracking optimization algorithm.

2.4. Contour constraints

Due to the fact that the normals from depth data are noisy at the boundaries, they do not constrain the movement of the template body enough. To overcome this issue, an energy functional that minimizes the point-to-plane distances between the silhouette of the depth map and that of the template model is proposed.

From our experiments, we concluded that computing the silhouette by means of the following approach yields good enough quality for the purpose of our application (see the graphical example in Fig. 4, left). For the depth map, a pixel is considered to be on the boundary if it has less than 7 neighbors in its 3×3 neighborhood with a small depth difference (we used 3 cm in our experiments). For the template mesh, the boundary vertices are detected by rendering the current pose of the mesh in a framebuffer and extracting them using morphological operators.

The energy functional for $E_{contour}$ is similar to the one in Eq. (2), with the differences that the correspondences are computed on the contour subsets as explained above, and the normals $\boldsymbol{n}_i$ (shown in green on the example contour image $\bar{\boldsymbol{I}}_{contour}$ in Fig. 4, right) are computed using the blurred contour image gradients as follows, with $\bar{\boldsymbol{I}}_{contour} = \boldsymbol{G}_{(\mu,\sigma)} \circ \boldsymbol{I}_{contour}$, and Gaussian kernel $\boldsymbol{G}_{(\mu,\sigma)}$:

$$\boldsymbol{n} = \frac{(\nabla_x \bar{\boldsymbol{I}}_{contour}, \nabla_y \bar{\boldsymbol{I}}_{contour}, 0)^T}{\left\| (\nabla_x \bar{\boldsymbol{I}}_{contour}, \nabla_y \bar{\boldsymbol{I}}_{contour}, 0) \right\|}. \qquad (3)$$

Furthermore, the correspondences are filtered by the angle between the projection of the template normals to the image plane and the depth pixel normals computed as above. Performing the normal computations and rejection step in 2D ensures more robustness to noisy input data.

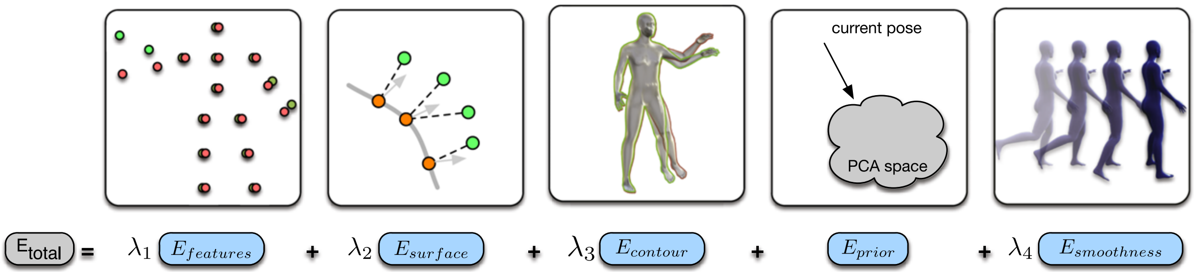

current pose
PCA space
$E_{total} = \lambda_1 E_{features} + \lambda_2 E_{surface} + \lambda_3 E_{contour} + E_{prior} + \lambda_4 E_{smoothness}$

Fig. 3. Visualization of the registration energies used in our optimization.

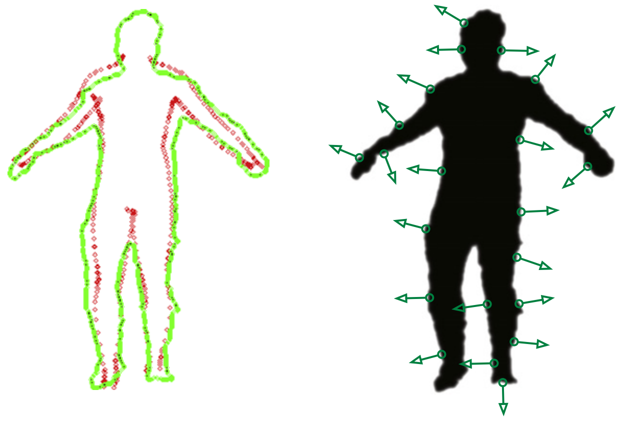


Fig. 4. Left: computed contour from depth and mesh: the green points represent the depth contour, the red points are the mesh contour, respectively. Right: associated normals computed on the depth map contour. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.5. Prior energy

Principal Component Analysis (PCA) is a dimensionality reduction technique that has been employed numerous times in the tasks of modeling [13,2] and tracking [14]. In particular, Douvantzis et al. [14] use PCA to decrease the number of variables needed to describe the pose of a human hand. On the one hand, by doing so, the optimization problem becomes easier due to less variables that lie within trained statistical boundaries. On the other hand, solutions can only be picked from within the learnt subspace, limiting the tracking algorithm to be able to follow only poses similar to those in the training set. Furthermore, human bodies can undergo more complex pose changes as compared to hands, and it is considered very difficult to generate a comprehensive training set that contains all possible human poses. As such, in order to allow for novel poses to be tracked while still penalizing unlikely poses, our approach uses the PCA subspace as a regularizer instead of an optimization space.

To begin with, we train the PCA model by using multiple long sequences tracked using the NiTE feature tracker. This tracker is imprecise, but enough to enable the generation of a large collection of plausible human poses. The covariance matrix of the de-meaned data matrix $\boldsymbol{D}$ obtained by concatenating rows of joint angles $\boldsymbol{\beta}_j$ for each frame j in the training set is expressed using eigenvalue decomposition as $\boldsymbol{C} = (\boldsymbol{D} - \boldsymbol{1}\boldsymbol{\mu})^T(\boldsymbol{D} - \boldsymbol{1}\boldsymbol{\mu}) = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{U}^{-1}$. $\boldsymbol{U}$ is the matrix formed of stacked columns of eigenvectors, $\boldsymbol{\Sigma}$ the eigenvalues in a diagonal matrix, $\boldsymbol{\mu}$ the mean of $\boldsymbol{\beta}$ over the training set. The eigenvectors are sorted in descending order by their corresponding eigenvalues and the first p modes are selected to form the PCA projection matrix $\boldsymbol{M}$. The number of modes p is chosen such that $\boldsymbol{M}$ forms a basis that explains a consistent portion of the training space (usually around 90%).

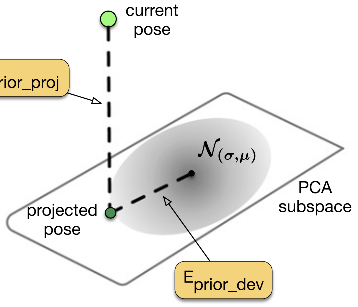


Fig. 5. Geometrical interpretation of the proposed PCA energy terms E_{prior_proj} and E_{prior_dev}.

In order to keep the estimated skeleton pose $\boldsymbol{\beta}$ in the feasible space of poses, we introduce an error term that measures the distance between $\boldsymbol{\beta}$ and the back-projection of its projection to the PCA space:

$$E_{prior\_proj} = \left\| (\boldsymbol{\beta} - \boldsymbol{\mu}) - \boldsymbol{M}\boldsymbol{M}^T(\boldsymbol{\beta} - \boldsymbol{\mu}) \right\|_2^2 . \tag{4}$$

The previous energy tries to push the current estimate of the angles close to their projection in the PCA space. While this gives a soft guarantee that the current estimate of $\boldsymbol{\beta}$ can be expressed as the linear basis learnt in the PCA model, it does not regularize the values to the variance seen in the training set. To this end, we introduce an additional energy that penalizes the distance of the projections of $\boldsymbol{\beta}$ to the mean of the PCA space. The projection along each subspace dimension is weighted by the inverse of the standard deviation of the corresponding PCA basis, which is the square root of the elements in the diagonal eigenvalue matrix $\boldsymbol{\Sigma}$:

$$E_{prior\_dev} = \left\| \boldsymbol{\Sigma}^{-1/2}\boldsymbol{M}^T(\boldsymbol{\beta} - \boldsymbol{\mu}) \right\|_2^2 . \tag{5}$$

Adding those two terms together, the prior energy functional with training becomes:

$$E_{prior} = \lambda_5 E_{prior\_proj} + \lambda_6 E_{prior\_dev} . \tag{6}$$

A geometrical interpretation of the two terms E_{prior_proj} and E_{prior_dev} is given in Fig. 5.

In the case a training set is not available for computing the PCA model, an alternative prior energy $\tilde{E}_{prior}$ is used to keep the skeleton close to its neutral A/T-pose:

$$\tilde{E}_{prior} = \|\boldsymbol{\beta}\|_2^2 . \tag{7}$$

However, it is important to point out that this method is less desirable than the data-driven technique described above as it allows for implausible human poses.

2.6. Smoothness energy

Tracking each frame independently leads to jitter due to high frequency differences between the values of $\boldsymbol{\beta}$ from one frame to another. To overcome this, we introduce a soft constraint that penalizes large jumps of $\boldsymbol{\beta}$ between consecutive frames:

$$E_{smoothness} = \left\| \boldsymbol{\beta}_{(t)} - \boldsymbol{\beta}_{(t-1)} \right\|_2^2 . \tag{8}$$

This is a first order smoothness term, enforcing that the angular velocity of the limbs is zero. More complex smoothness terms could be used in order to regularize the problem using acceleration or even higher order derivatives of the pose vector. We deemed the energy in Eq. (8) to be sufficient for the purpose of this system, as we do not expect excessively fast motions for the scenario of dynamic body scanning.

The smoothness energy term could be omitted, and the sequence could be smoothed as a post-processing stage, via temporal Laplacian filtering, for example. However, the smoothness energy included into the optimization acts as a soft prior term, adjusting the angular velocities of the limbs, and attracting the variables to the solution of the previous frame.

2.7. Tracking and modeling

The optimization is split into two stages: *tracking*, aimed at estimating the pose $\boldsymbol{\beta}_i$ of the skeleton for each frame i, and *modeling*, aimed at estimating the body shape parameters $\boldsymbol{s}$ over the whole sequence. These two stages are briefly outlined in the following.

Tracking The tracking is performed sequentially for each frame, by keeping $\boldsymbol{s}$ fixed and finding values of $\boldsymbol{\beta}$ for which E_{total} is minimized. The Levenberg–Marquardt algorithm is used to optimize for the tracking, in which the linear system to be solved is computed analytically. The adaptive damping technique present in this algorithm is needed as the problem is unstable because of the chain of variables that influences all the nodes below in the tree. If the smoothness term $E_{smoothness}$ is removed from the total energy, then the tracking can be solved for each frame independently, allowing for heavy parallelization of this stage of the pipeline.

Modeling As mentioned before, the modeling stage refers to finding the optimal blendshape weights $\boldsymbol{s}$ such that the mesh vertices $\boldsymbol{v}_i^{rest} = \boldsymbol{m}_i + \boldsymbol{B}_i\boldsymbol{s}$ minimize the modeling error over the sequence of frames tracked so far. Note that the modeling error does not contain some of the terms in E_{total}, as those were pertaining to regularizing the solutions for $\boldsymbol{\beta}$:

$$E_{modeling} = \gamma_1 E_{surface} + \gamma_2 E_{contour} + \gamma_3 E_{bs\_reg} . \tag{9}$$

The tracking and modeling paradigm we proposed can be used for both online and offline modeling. Similar to [15], one option is to use a temporal weighting scheme for the accumulation of the per frame constraints, giving more weight to more recent frames. In such a scheme, the modeling is done after each tracked frame, continuously updating the body model during the tracking. This is expressed mathematically as solving the linear system at time t, $\boldsymbol{M}_{lhs}^t \Delta\boldsymbol{s}^t = \boldsymbol{M}_{rhs}^t$. The update is done as follows, with:

$$\boldsymbol{J}^t = \frac{\partial E_{modeling}^t}{\partial \boldsymbol{s}^t} \tag{10}$$

$$\boldsymbol{b}^t = E_{modeling}^t \tag{11}$$

the left and right-hand-side of the linearized constraints for frame t, respectively; w_γ^t is the temporal weight determined by the parameter $\gamma < 1$, which quantifies the influence of recent frames in the detriment of old frames:

$$w_\gamma^t = \gamma w_\gamma^{t-1} + 1 \tag{12}$$

$$\boldsymbol{M}_{lhs}^t = \gamma \frac{w_\gamma^{t-1}}{w_\gamma^t} \boldsymbol{M}_{lhs}^{t-1} + \frac{1}{w_\gamma^t} (\boldsymbol{J}^t)^T \boldsymbol{J}^t \tag{13}$$

$$\boldsymbol{M}_{rhs}^t = \gamma \frac{w_\gamma^{t-1}}{w_\gamma^t} \boldsymbol{M}_{rhs}^{t-1} + \frac{1}{w_\gamma^t} (\boldsymbol{J}^t)^T \boldsymbol{b}^t . \tag{14}$$

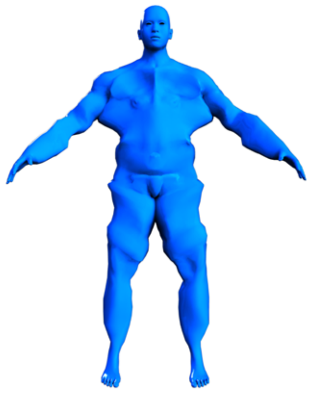


Fig. 6. Unrealistic estimated body model due to not regularizing the blendshape weights $\boldsymbol{s}$.

The second option is to accumulate all the frames of the sequence with equal weights ($\gamma = 1$ in the equations above), and solve for the modeling only at the end of the tracked input sequence. The main disadvantage of this scheme is that multiple tracking passes of the sequence are required, but the results will be more consistent with the whole dataset.

Allowing for any values of $\boldsymbol{s}$ can lead to unrealistic models such as the one depicted in Fig. 6. Regularization is needed to keep the optimization from converging to such unwanted solutions. To this end, we suggest two options:

$$E_{bs\_reg\_L2} = \|\boldsymbol{s}\|_2^2 \tag{15}$$

$$E_{bs\_reg\_L1} = \|\boldsymbol{s}\|_1 . \tag{16}$$

The effect of using either one of these regularization energies is explained in Fig. 7. By employing $E_{bs\_reg\_L2}$, the linear system can be solved with the Gauss–Newton solver, with no damping necessary. Using the L1 norm present in $E_{bs\_reg\_L1}$ necessitates a different solver. In our implementation we use the Gauss–Seidel method with successive over-relaxation adapted to the L1 norm regularization [16], along with iterative reprojections to keep the blendshape weights in the [0, 1] range. Another option, albeit usually considered slower, would have been Gauss–Newton with iterative reweighting [17].

In some situations the tracking does fail, and using those wrong constraints for the modeling phase might lead to erroneous results. In order to avoid such situations, we skip the frames for which the global tracking error E_{total} is higher than a certain threshold and the percentage of the overlap between the depth map and the template model surface is below a certain value. (in our experiments we chose 45%).

3. Implementation

The skeleton is scaled once at the beginning of the pipeline by taking the median distances between the NiTe features at each frame over the whole sequence as being the limb lengths. From our experiments, this proved to be sufficient, and no further limb size adaptation was necessary during the tracking refinement and modeling iterations. Such a solution would not be possible in the case of online modeling and a different heuristic for the limb size adaptation should be employed.

The current implementation of the framework uses PCL exclusively, by relying on multiple components for the pre-processing, correspondence estimation and visualization stages. In particular, for efficient Nearest Neighbor search in the point cloud 3D domain we rely on the FLANN library included in PCL. Normals on the depth maps and on the point clouds are estimated, respectively, with the

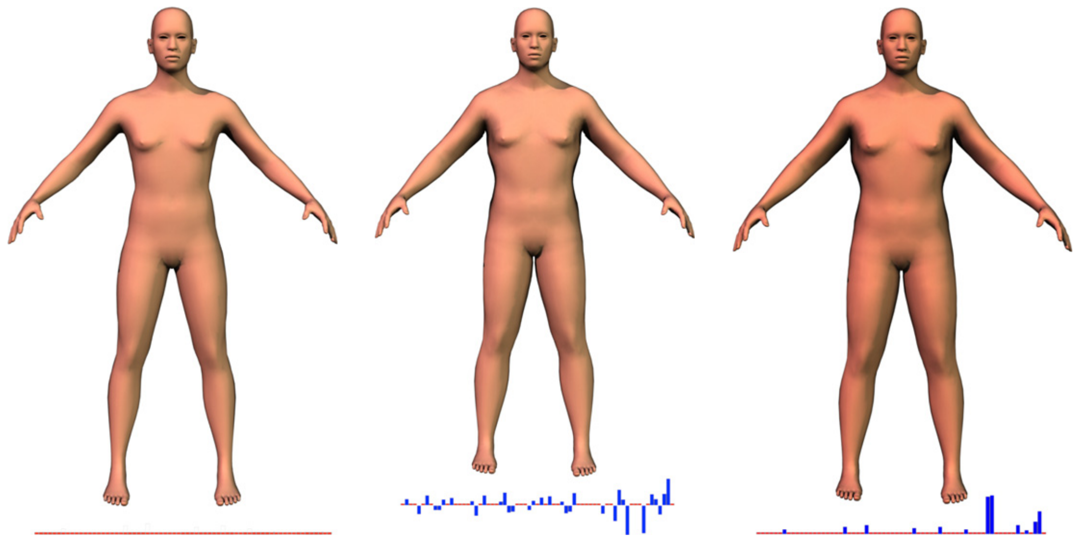

(a) Template model avg. tracking error: 0.39. (b) L2 fitting avg. tracking error: 0.32. (c) L1 fitting avg. tracking error: 0.34.

Fig. 7. Visualization of the effect of the different norms on the blendshape weights for the body modeling of subject S4. Displayed are the resulting body meshes, along with a graphical representation of the blendshape weights and the average tracking error across the whole depth sequence obtained by using the respective body model estimation. (a) shows the scaled template model. (b) shows the modeling result with the classical L2 norm energy on the blendshape weights. (c) showcases the body model obtained by regularizing the blendshape weights with an L1 norm. The tracking error becomes slightly higher, but notice the smaller number of activated blendshapes with high values, compared to the L2 fitting, where a lot of blendshapes are activated with small values.

multithread method *pcl::NormalEstimationOMP* and with the integral images-based method *pcl::IntegralImageNormalEstimation*. All the solvers are implemented using the Eigen C++ linear algebra library, with no other external dependencies. For parallelization, we used the OpenMP library.

We have performed all our experiments on data collected from an Asus Xtion PRO LIVE sensor, which consists of synchronized depth and color images at a resolution of 640×480 each, delivered at 30 Hz. For the performed experiments, the blendshape model used for fitting contained 18 meshes selected from MakeHuman, representing macro shape variations of the body. The template mesh has 13 380 vertices and 26 756 triangles. The skeleton structure used is the *second_life* rig provided in the MakeHuman application, which has been manually mapped to the NiTE tracking features. The pose prior has been trained from a database of about 2300 frames with 18 3D feature locations per frame. The compressed PCA model retains 91% of the variability of the motion database by using 12 modes.

4. Experimental results

In this section, we provide some experimental results and applications of our tracking and body shape estimation framework. To this goal, we have acquired multiple sequences of around 500–600 frames for each person of a group of four males and two female subjects (subjects $S1 \cdots S6$). Fig. 8 shows one frame relatively from each acquired subject, together with the corresponding estimated body model. As it can be seen, the setting is that of a typical indoor environment, which includes cluttered background. As witnessed by the figure, the estimated body models are different to fit the specific body traits of each subject. In addition, we show, in Fig. 9, multiple examples of tracked and estimated body models for different sequences, along with the corresponding RGB frames. As it can be seen, the method can track the 3D body also in complex poses, and also when the person's back is facing the camera. For a more clear view of the incremental tracking and modeling process, Fig. 10 shows the evolution of the template silhouette with respect to the depth map contour as the optimization converges.

To evaluate quantitatively the modeling precision of our framework, we computed the standard deviations measured on a few key locations of the body model estimated along each of the 4 sequences associated to each of the 4 subjects for which such number of sequences was available (i.e., subjects $S1 \cdots S4$). The results are reported in Table 1, along with a visualization of the location of the body features (on the left). The reported results range between a minimum of 0.36 cm and a maximum of 2.35 cm, with an average over all locations and all subjects of 1.10 cm, which demonstrates an encouraging repeatability of the proposed modeling algorithm, leading us to believe that this framework has the potential to obtain relatively accurate results, even without the need of a complicated setup or scripted actor movement.

We wish to point out here that our framework does not explicitly take into account the presence of clothes. Indeed, precise body pose measurements should be taken without clothes or wearing garments that are tight to the body. Indeed, loose clothes might easily lead to errors in both the tracking and modeling stages of the pipeline, thus resulting in decreasing the accuracy of body measurements. The same problem would occur if the subject has long/voluminous hair, or in presence of any accessory with a non negligible size, such as bags, hats, glasses, belts, etc.

As mentioned before, the L1 semantic body modeling we propose opens up multiple avenues for applications that could have not been possible with L2 statistical models. A simple such example is depicted in Fig. 11, where the word clouds corresponding to the blendshape weights of the body fitted in Fig. 7 have been created. Note the compactness of the semantic representation yielded by the use of the L1 relaxation with respect to the one yielded by the L2 model.

Moreover, by having the body mesh modeled and tracked throughout the sequence, and being the associated RGB frame available as well in correspondence with each depth frame (i.e., RGB-D data), our framework allows to build a complete texture of the mesh (see Fig. 12). A simple technique is employed that projects each RGB frame into the UV-space of the mesh,

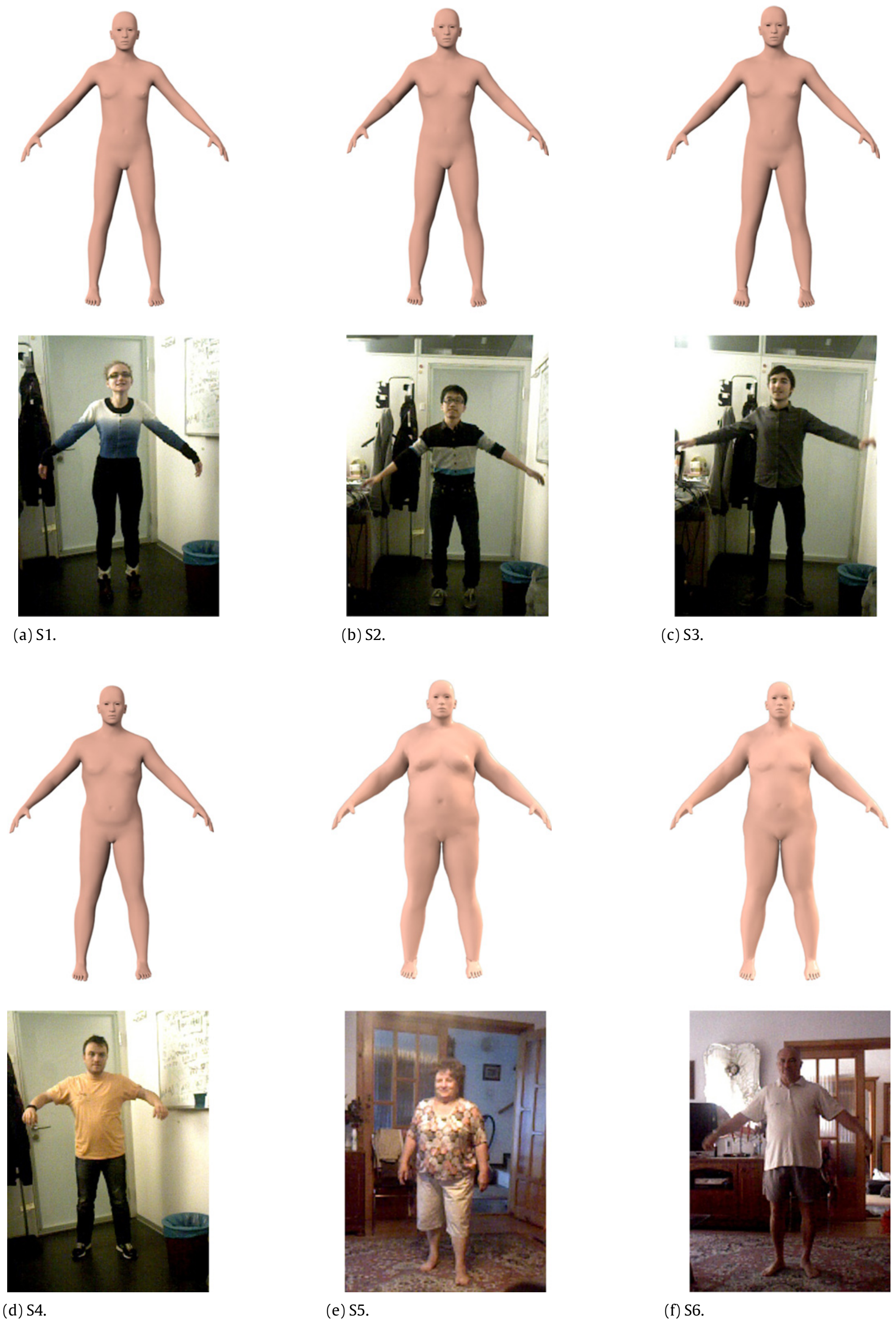


Fig. 8. Modeling results showing the six experimental subjects ($S1 \cdots S6$) together with their modeled 3D body shape.

accumulating color contributions weighted by the foreshortening angle (angle between the normal at the mesh surface and the viewing direction of the camera). Note the presence of artifacts such as blurring and black regions. These are due to inaccuracies in the tracking and the fact there are regions that have not been captured during the sequence.

4.1. *Effect of each tracking energy*

Finally, in Fig. 13 we show some results relatively to the influence of the most relevant energy terms employed in the proposed cost function, obtained by deactivating them one at a time and highlighting the most remarkable qualitative differences.

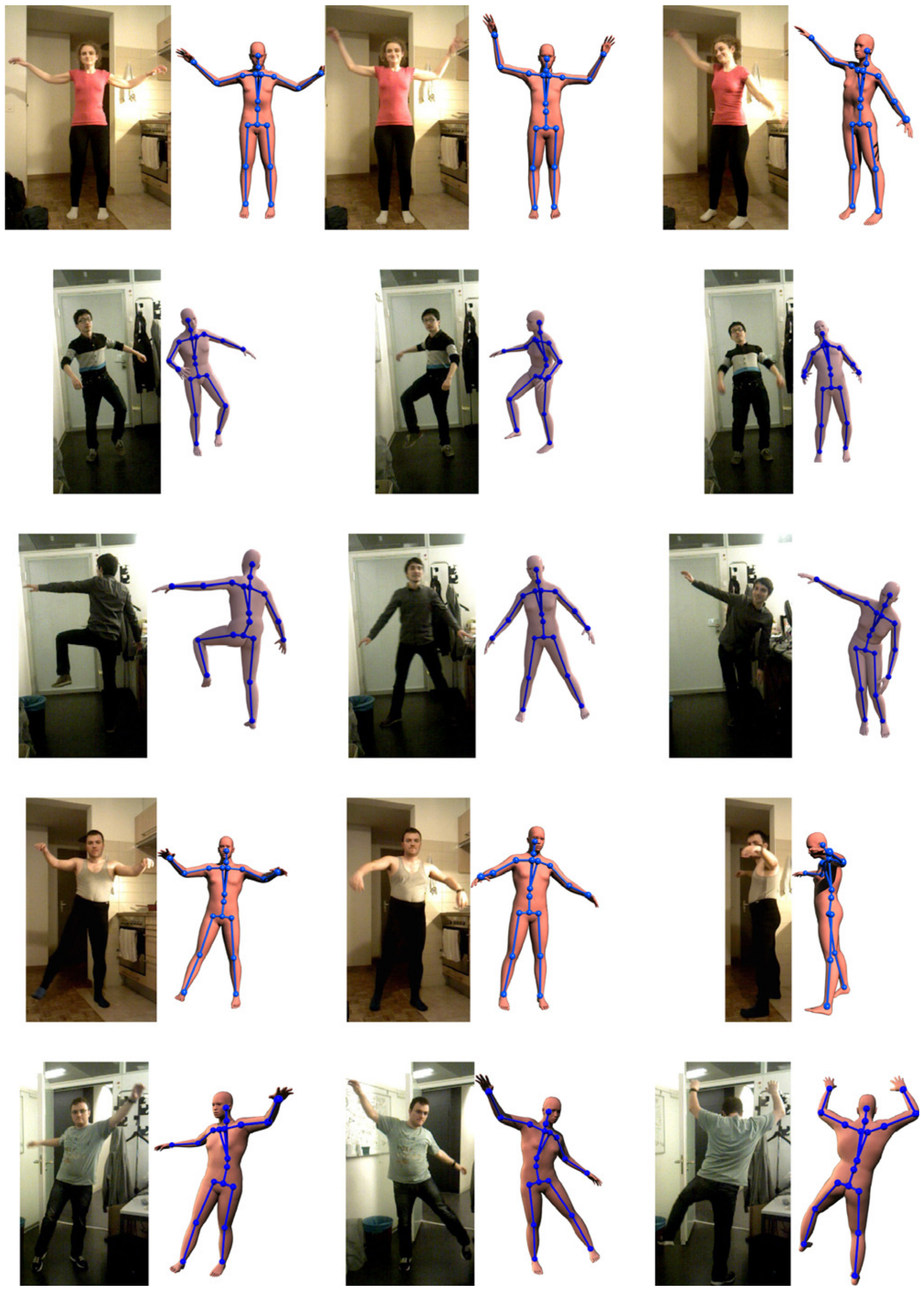


Fig. 9. The proposed framework is able to track bodies and estimate their parametric body model from a depth sequence. In each row of the figure, we report three examples of a sequence corresponding to one of the evaluated subjects (S1, S2, S3, S4, and S4, respectively). For each example, we show the input RGB frames, the posed body and its corresponding skeleton.

Table 1
Mean and average standard deviation of all the measurements for 4 test subjects with 4 recorded sequences each. All reported values are in cm. On the left: locations of the five measurements taken on the subjects.

	S_1		S_2		S_3		S_4	
	μ	σ	μ	σ	μ	σ	μ	σ
arm	25.97	0.35	28.95	0.42	29.4	0.36	32.55	1.05
chest	89.59	0.70	100.96	2.35	105.35	0.6	108.68	1.01
hips	92.27	1.24	99.37	1.01	105.21	0.59	106.65	0.67
leg	48.74	1.07	54.91	0.49	55.6	0.91	59.56	1.22
waist	76.70	1.53	83.2	1.92	91.25	0.84	94.67	1.12
avg		1.06		1.46		0.69		1.05
total avg σ	1.10							

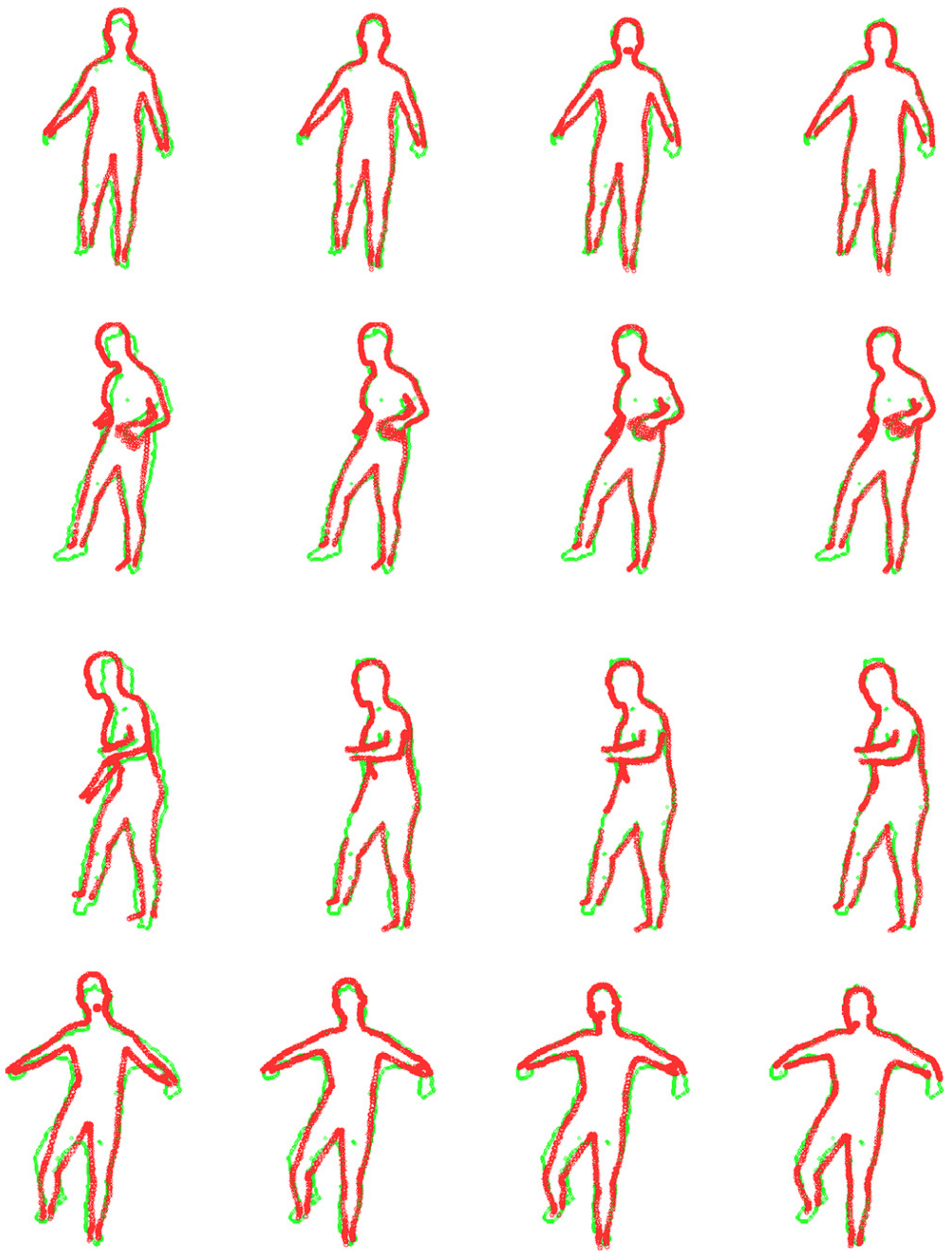


Fig. 10. Each row of images shows the progress of the tracking and modeling optimization for the contours of single frames of actor S4, starting from the initial estimate given by the 3D feature locations and the template mesh, up to convergence for both the pose and the body shape parameters. The template mesh silhouette is drawn in red, and the depth map contour is green. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

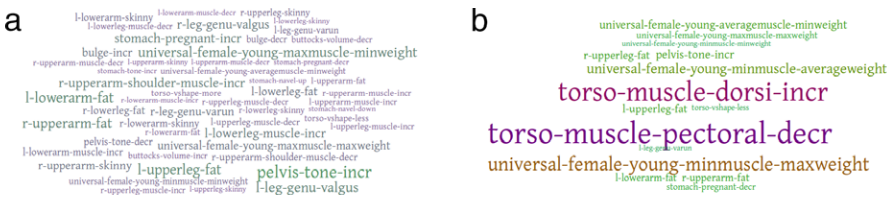


Fig. 11. Example application of the proposed semantic body modeling. (a), (b) show word clouds created using the activated blendshape names and their weights for the L2 blendshapes regularization, and the L1, respectively.

As witnessed by Fig. 13(a), the **surface registration energy** encapsules the most important set of constraints of the proposed optimization. Its purpose is to align the underlying scene surface sampled by the depth sensor with the template mesh. The contour and feature constraints in addition to the tracking priors are not enough for precise tracking. Differently, the **contour energy** is useful because the depth map normals are rather flat at the silhouette of the objects in the scene, thus not constraining the tracking enough in those regions. As such, misalignments like the one in Fig. 13(c) can occur as the point to plane energy is minimized even if the hand alignment is off. The contour correspondences shown in the zoomed in part of the figure would have pulled the

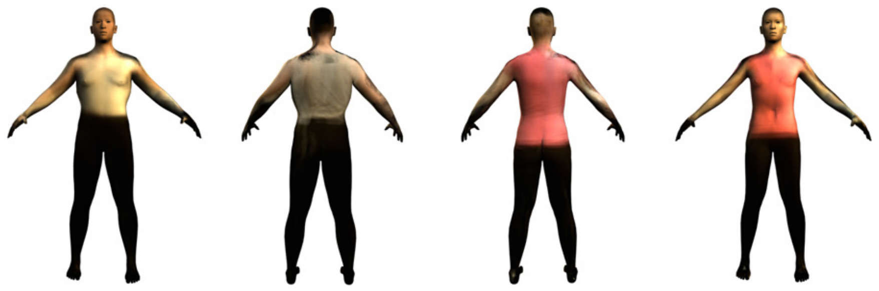
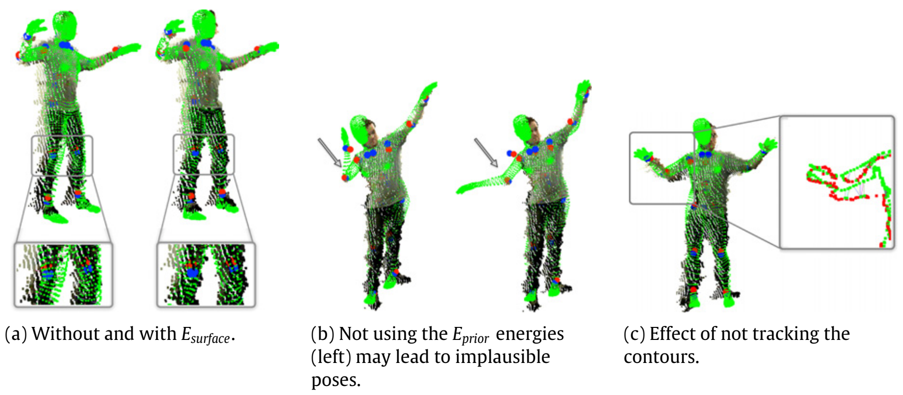


Fig. 12. Our framework allows for texturing the body meshes using a weighted average of the color contributions from each RGB frame. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)



(a) Without and with $E_{surface}$. (b) Not using the E_{prior} energies (left) may lead to implausible poses. (c) Effect of not tracking the contours.

Fig. 13. Influence of the proposed energy terms on the resulting tracking.

Table 2
Performance benchmarking results for each stage of the pipeline. The average number of iterations is per frame. The rest of the total per frame average time is spent with other book-keeping operations.

Step	Average number of iterations	Average time (ms)
Normal estimation	1	24
Initialize IK with 3D Features	1	1
Point to plane correspondences	3.5	3
Contour correspondences	3.5	26
IK with all energies	3.5	27
Accumulate modeling constraints	1	3
Tracking per frame	–	279
Modeling per frame	–	3

template body to its correct location. These effects can accumulate in time and lead to a complete loss of tracking.

Moreover, the **prior energy** is needed in order to avoid implausible poses to be outputted. In the situations when the input data is lacking information about certain regions of the body due to occlusions, the statistical pose priors we propose help keep the body in a reasonable pose. An example is shown in Fig. 13(b), where the 3D features were wrong. Without priors the tracker moves the arm in an impossible pose, but employing the priors (right side) converged to a more plausible solution. Finally, although not shown in the Figure, the **feature energy** that keeps the joint position close to the NiTE 3D features detected for each frame is especially useful for initial alignment, after which its weight can be decreased to zero during the optimization. Without it, manual intervention would be mandatory to pose the body so that the iterative optimization tracking can have a warm start.

4.2. Performance evaluation

We have performed our experiments on a laptop with an 8-core Intel Core i7-4940MX processor, with 32 GB of RAM, and a GeForce GTX 880 M graphics card, running Ubuntu 14.10. The code uses the CPU for all of the computation, with the exception of the framebuffer rendering for extracting the mesh silhouette, which is done using primitive OpenGL calls. Table 2 collects the timing information from our experiments. Furthermore, it is worth mentioning that the IK optimization using only the 3D features was tuned to use an average number of 24.1 Levenberg Marquardt iterations, and the IK optimization with all the energies performed with an average of 10.44 internal iterations.

5. Conluding remarks

In this paper we have presented a modular framework for 3D body tracking and parametric modeling. Our framework can run efficiently, and thanks to the use of MakeHuman models together with L1 relaxation allows for new applications such as semantic body part tagging of the acquired subjects, as well as body model texturing. Moreover, our framework can be easily adapted to fitting posed bodies that have been scanned using technologies such as PCL's Kinect Fusion [18] implementation, KinFu, or multiview reconstruction using external tools such as 123DCatch [19].

The code is modularized in a logical structure that allows for further experimentation and extensions. We are planning on releasing the code as open-source, and integrate it in the *pcl::bodies* module of the PCL library. Indeed, we believe that our contribution can lead to a higher interest from developers and researchers to dive into more complex body tracking and modeling applications. Also, we believe that due to its customizability, it will enable future work which includes novel research, better benchmarking data and interesting new applications.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at http://dx.doi.org/10.1016/j.robot.2015.09.029.

References

[1] B. Allen, B. Curless, Z. Popović, The space of human body shapes: reconstruction and parameterization from range scans, ACM Trans. Graph. (TOG) 22 (3) (2003) 587–594.
[2] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, J. Davis, Scape: Shape completion and animation of people, ACM Trans. Graph. (TOG) 24 (3) (2005) 408–416.
[3] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, H.-P. Seidel, A statistical model of human pose and body shape, Comput. Graph. Forum 28 (2) (2009) 337–346.
[4] M. Bastioni, S. Re, S. Misra, Ideas and methods for modeling 3D human figures: The principal algorit hms used by makehuman and their implementation in a new approach to parametric modeling, in: Proceedings of the 1st Bangalore Annual Compute Conference, ACM, 2008, p. 10.
[5] P. Guan, A. Weiss, A.O. Balan, M.J. Black, Estimating human shape and pose from a single image, in: 2009 IEEE 12th International Conference on Computer Vision, IEEE, 2009, pp. 1381–1388.
[6] A. Weiss, D. Hirshberg, M.J. Black, Home 3D body scans from noisy image and range data, in: 2011 IEEE International Conference on Computer Vision (ICCV), IEEE, 2011, pp. 1951–1958.
[7] C. Malleson, M. Klaudiny, A. Hilton, J.-Y. Guillemaut, Single-view RGBD-based reconstruction of dynamic human geometry, in: 2013 IEEE International Conference on Computer Vision Workshops (ICCVW), IEEE, 2013, pp. 307–314.
[8] T. Helten, A. Baak, G. Bharaj, M. Muller, H. Seidel, C. Theobalt, Personalization and evaluation of a real-time depth-based full body tracker, in: Proc. Int. Conf. on 3D Vision, 3DV, 2013.
[9] M. Ye, R. Yang, Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera, in: Proc. Int. Conf. on Computer Vision and Pattern Recognition, CVPR, 2014.
[10] M. Munaro, A. Basso, A. Fossati, L. Van Gool, E. Menegatti, 3D reconstruction of freely moving persons for re-identification with a depth sensor, in: 2014 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2014, pp. 4512–4519.
[11] S. Holzer, R.B. Rusu, M. Dixon, S. Gedikli, N. Navab, Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images, in: 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2012, pp. 2684–2689.
[12] S. Rusinkiewicz, M. Levoy, Efficient variants of the ICP algorithm, in: 2001. Proceedings. Third International Conference on 3-D Digital Imaging and Modeling, IEEE, 2001, pp. 145–152.
[13] V. Blanz, T. Vetter, A morphable model for the synthesis of 3D faces, in: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press/Addison-Wesley Publishing Co., 1999, pp. 187–194.
[14] P. Douvantzis, I. Oikonomidis, N. Kyriazis, A. Argyros, Dimensionality reduction for efficient single frame hand pose estimation, in: Computer Vision Systems, Springer, 2013, pp. 143–152.
[15] S. Bouaziz, Y. Wang, M. Pauly, Online modeling for realtime facial animation, ACM Trans. Graph. (TOG) 32 (4) (2013) 40.
[16] W.J. Fu, Penalized regressions: The bridge versus the lasso, J. Comput. Graph. Statist. 7 (3) (1998) 397–416.
[17] R. Chartrand, W. Yin, Iteratively reweighted algorithms for compressive sensing, in: 2008. ICASSP 2008. IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2008, pp. 3869–3872.
[18] R.A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A.J. Davison, P. Kohi, J. Shotton, S. Hodges, A. Fitzgibbon, Kinectfusion: Real-time dense surface mapping and tracking, in: 2011 10th IEEE International Symposium on Mixed and Augmented Reality (ISMAR), IEEE, 2011, pp. 127–136.
[19] J. Biehler, B. Fane, 3D Printing with Autodesk: Create and Print 3D Objects with 123D, AutoCAD and Inventor, first ed., Que Publishing Company, 2014.

Alexandru E. Ichim is currently a Ph.D. student at Ecole Polytechnique Federale de Lausanne, in the Computer Graphics and Geometry Laboratory, as well as a Point Cloud Library (PCL) developer and a Scientist for the Open Perception foundation. He obtained his Bachelor of Science in 2011 at Jacobs University in Bremen, Germany, and his M.Sc. in 2013 from EPFL. During his studies he collaborated with institutions such as the German Research Center for Artificial Intelligence (Robotics Innovation Center and Cognitive Systems), Jacobs Robotics Group, EPFL, and Willow Garage. His current research focuses on human reconstruction and performance capture.

Federico Tombari holds an appointment as an Assistant Professor at the University of Bologna since October 2013, after obtaining from the same institution a Ph.D. in 2009. Since September 2014, he has also been a Guest Professor at the Computer Aided Medical Procedures (CAMP) Chair at the Technical University of Munich (TUM). His current research activity concerns computer vision and robot perception. It encompasses co-authoring more than 60 refereed papers on peer-reviewed international conferences and journals, mainly focused on visual data representation, object recognition, stereo vision, video analysis for surveillance and 3D perception. In 2008 he was an intern at Willow Garage, California. He is a Senior Scientist volunteer for the Open Perception foundation and a developer for the Point Cloud Library, contributing also in terms of dissemination and mentoring for code sprints sponsored by private companies (Google, Trimble, Honda). In 2014 he served as administrator and mentor for PCL in the Google Summer of Code. He is member of IEEE and IAPR-GIRPR.