Fast deformable model-based human performance capture and FVV using consumer-grade RGB-D sensors

Dimitrios S. Alexiadis, Nikolaos Zioulis, Dimitrios Zarpalas and Petros Daras

Centre for Research and Technology-Hellas,
Information Technologies Institute,
6th km Charilaou-Thermi, Thessaloniki, GR-57001, Greece,
e-mail: dalexiad@iti.gr; nziolus@iti.gr; zarpalas@iti.gr; daras@iti.gr

Abstract

In this paper, a novel end-to-end system for the fast reconstruction of human actor performances into 3D mesh sequences is proposed, using the input from a small set of consumer-grade RGB-Depth sensors. The proposed framework, by offline pre-reconstructing and employing a deformable actor's 3D model to constrain the on-line reconstruction process, implicitly tracks the human motion. Handling non-rigid deformation of the 3D surface and applying appropriate texture mapping, it finally produces a dynamic sequence of temporally-coherent textured meshes, enabling realistic Free Viewpoint Video (FVV). Given the noisy input from a small set of low-cost sensors, the focus is on the fast ("quick-post"), robust and fully-automatic performance reconstruction. Apart from integrating existing ideas into a complete end-to-end system, which is itself a challenging task, several novel technical advances contribute to the speed, robustness and fidelity of the system, including a layered approach for model-based pose tracking, the definition and use of sophisticated energy functions, parallelizable on the GPU, as well as

a new texture mapping scheme. The experimental results on a large number of challenging sequences, and comparisons with model-based and model-free approaches, demonstrate the efficiency of the proposed approach.
Keywords: Dynamic mesh sequences, RGB-D sensors, 3D reconstruction, articulated motion, surface deformation, deformable models, Free Viewpoint

1. Introduction

With the convergence of technologies from computer vision, computer graphics, multimedia and relevant fields, the development of new media types, such as Free Viewpoint Video (FVV) [1], is now enabled. In this work, we deal with 360° FVV of human performance, where the 3D content is modelled on the basis of 3D textured meshes, since this is the only option that enables interactive full-3D experience from a small set of capturing cameras [1]. This selection however introduces the need for robust and accurate 3D reconstruction, which is a challenging task.

While recent FVV systems provide high quality results [2], apart from being quite slow, they require very expensive setups. In this paper, a low-cost, end-to-end system for human performance reconstruction, is described. We focus on the fast ("quick-post"), realistic reconstruction using consumer-grade RGB-D sensors, which introduce several challenges, such as noisy and unsynchronized input. More importantly, we target time-coherent 3D reconstructions, and specifically dynamic mesh sequences, i.e. meshes with constant vertex/face count and connectivity across frames, which enable their efficient compression, storage and streaming [3], in contrast to Time-Varying Meshes (TVM) [4]. Towards this end, a novel deformable model-based recon-

struction system is proposed, i.e. a system that exploits a-priori knowledge about the captured subject to parameterize and constrain the reconstruction process. The method relies on the automatic off-line generation of the actor's model and on-line human pose tracking and surface deformation to finely match the captured data. Implicitly, the method tracks robustly the skeletal user's motion. Although the idea of model-based reconstruction is not new, to the authors knowledge, the proposed system is among the first few **model-based** ones that make use of low-cost RGB-D sensors. It can reconstruct faithfully and fully-automatically the surface details, as well as texture; while, it performs faster than state-of-art (SoA) model-based methods that use multiple passive or active cameras.

Given the dynamic (vs time-varying) nature of the output mesh sequence, efficiently encoding/decoding the data is possible (e.g. [3]). Although the proposed reconstruction method does not generate meshes in real-time framerates, the efficient decoding of the output mesh data and the use of OpenGL shaders for 3D rendering in real-time is straightforward, i.e. the output data of the method is appropriate for real-time FVV applications.

A major contribution of the current work is the appropriate selection, combination and integration of existing ideas into a complete end-to-end system. Additionally, novel technical contributions are proposed, in order to achieve the given objectives, including: i) The introduction of a 3-layers framework, where the first layer is fast and each next layer (executed only when needed) is slower, but more robust; This layered approach introduces speed and robustness in pose tracking; ii) Instead of modeling human joints as a concatenation of 1-DOF joints, a full 3-DOF model is exploited, which

lets also the implicit application of valid human-pose constraints; iii) A set of sophisticated energy functions, fast computable and parallelizable on the GPU is proposed; iv) The final surface, deformed iteratively via a Laplacian framework, is textured taking advantage of the dynamic nature of the output meshes.

The rest of paper is organized as follows: In subsection 1.1, the related SoA approaches are presented, discussing also the differences with the proposed one. Section 2 gives an overview of the proposed framework, while sections 3, 4 and 5 provide all the necessary theoretical and technical details. The experimental results on a large number of sequences and comparisons vs relevant recent approaches, presented in section 6, demonstrate the efficiency of the proposed approach.

1.1. Related work

Initially, a short survey on per-frame (static) 3D reconstruction from multiple sensors is given; Kinect-based systems for time-varying reconstruction are then reviewed, before focusing into the more relevant model-based methods.

Per-frame (static) reconstruction from multiple sensors

Although several passive RGB camera-based methods can be found, e.g. [5] and [6], due to their quite slow performance, they are mainly used for the off-line reconstruction of static objects. Other simpler methods are often preferred, such as Shape-from-Silhouette (SfS) methods [7, 8], but their quality may suffer, especially with a small number of cameras and without a green-screen background for robust silhouette extraction. To cast such methods more robust to inconsistent silhouettes, errors in background subtraction

etc., the 3D shape recovery problem from the available noisy silhouettes is often formulated as a continuous energy minimization problem that employs regularization terms, as e.g. in [9], increasing significantly however the computational effort.

Space-carving or voxel-coloring techniques [10, 11], although potentially more accurate, are generally less robust, since they are based on making irreversible decisions on voxels' removal. A semi-supervised SfS approach is described in [12], which employs a user-defined contour as a starting point to obtain "probabilistic" silhouettes of successive images; thus obtaining a prior distribution for the probability of a voxel being carved. The method combines the advantages of SfS techniques and statistical space-carving approaches.

Finally, the idea of multi-view stereo has also been much investigated [13, 14, 15, 16], with [13] having pioneered FVV research with the "Virtualized Reality" system, using a dome of 51 synchronized cameras.

Instead of acquiring the depth information from passive stereo pairs, active direct-ranging sensors can been used. Relevant methods have been early proposed for the fusion of range data [17, 18], to produce, off-line, a single mesh. Due to their robustness against noise, recent explicit-fusion methods [19] or implicit-fusion volumetric methods, such as the Poisson [20] and its ancestor Fourier Transform (FT)-based method [21], are worth mentioning here.

The referenced methods work on a per-frame basis and the production of time-coherent meshes is possible only with post-processing on top of such methods [22].

Kinect-based systems for TVM reconstruction

As the technology of real-time structured light-based [23] and Time-Of-Flight sensors is getting mature and with the appearance of consumer-grade RGB-Depth cameras, variations of the referenced active sensor-based approaches [17, 18, 21] have been employed, targeting mainly real-time 360^o reconstruction for tele-presence applications [24, 25, 26]. Such methods, working on a per-frame basis, do not produce time-coherent meshes.

In another class of Kinect-based systems, the information in multiple consecutive frames is fused to produce smooth reconstructions using a single sensor. With optimized GPU code, real-time operation is achieved: The recent and promising "Dynamic Fusion" (DynFu) work by [27] can reconstruct slowly deforming objects (e.g. slowly moving upper body) in a real-time SLAM framework, where apart from the camera pose, a volumetric 6D motion field is also recovered. The VolumeDeform system [28] improves the DynFu idea, by combining the dense depth-based constraints with the extraction and use of sparse RGB-based 2D features. These methods obtain remarkable results, given the noisy Kinect input. Nevertheless, they are applied with a single sensor at close distances, for relatively slow motions, while they do not address the texture mapping problem, and produce TVM sequences.

Model-based methods

Apart from the depth cameras-based methods that are presented later in this subsection, most model-based methods make use of passive RGB cameras. Below, we differentiate between pure motion capture methods, i.e. methods that focus only in pose estimation, and full "performance capture" methods that additionally handle highly non-rigid deformations.

Human pose estimation and tracking: Among the first works on model-based FVV of humans, Carranza et al. [29] extract the actor's silhouettes in 8 synchronized views and recover the model's pose sequentially (i.e. initially the torso and then the limbs) by minimizing a silhouettes-based energy. Although the "puppeted" (skeleton-based deformed) model is only roughly aligned with the silhouettes, an accompanying texture-mapping approach is also proposed.

A survey on model-based pose estimation can be found in [30]. Bregler et al. [31] demonstrated for the first time a technique that is able to track articulated human body configurations, by integrating the notion of "twists" from the robotics community [32] into differential optical-flow. Many relevant SoA methods [33, 34, 35, 36], which track the human pose using Inverse Kinematics (IK) and make use of "twists", are correspondence - based, i.e. they are based on the key idea of collecting correspondences between 3D points of the model and 2D points on the image observations.

Gall et al. [36, 37] propose the use of a particle-based, global optimization approach, well suited to the problem of skeletal human motion, the Interacting Simulated Annealing (ISA) [38]. An energy function with silhouette, appearance and physical constraints is optimized in [36]. Their ISA-based approach, although very slow, provides robust skeletal motion tracking. Similar annealed particle filter-based methods for human pose estimation tracking have also been proposed, e.g. [39].

Brox et al. [33] propose the combined use of 2D contour-based correspondences and region fitting with dense optical flow and SIFT features to handle large transformations. Corazza et al. [34], instead of searching for correspondences on the 2D plane, construct the visual hull and collect 3D

correspondences with the model, through an articulated ICP.

Finally, a sub-class of pose estimation methods, which are based on learning from large training sets, should be also mentioned here [40, 41, 42]. For example, the work in [40] reconstructs the 3D body pose from image sequences based on top-down learning: In the learning stage, the body model database is constructed by classifying the training data into sub-clusters with silhouette images, while in the reconstruction stage, the cluster for the best matching silhouette image is searched using silhouette history images. In the training phase of [41], the human motion subspace is extracted by performing conventional PCA on a sample set of motion capture data, so that to reduce the problem dimensionality and make the pose recovery process more effective. For the pose-recovery phase, an annealed genetic algorithm is used to search the optimal human-pose solution. In [42], a number of "activity models" is defined, each learnt for a particular class of human activity. A "multiple activity model annealed particle filtering (MAM-APF)" scheme is then proposed to efficiently combine of these learnt activity models.

Full "performace capture": In the work of Vlasic et al. [43], the visual-hull is initially reconstructed and the human pose is recovered by minimizing an objective function that incorporates the signed distance of the skeleton bones from the visual-hull surface, a temporal smoothness term, and the distance of end-effectors from the extremities. As stated by the authors, the tracking system may fail on complex motions. In a second stage, refinement of the "puppeted" model is realized, using a Laplacial deformation framework [44, 45]. The target positions of model's silhouette rim vertices are obtained by matching with the silhouette contours.

Gall et al. [35] propose the use of a 2-Layers approach for pose tracking. When the first layer fails, the second layer is used to recover the pose of misaligned limbs. The 1st layer collects a sufficient set of 3D model-to-2D correspondences, relying on silhouette contours and texture information. The slower 2nd layer, relying on ISA-based optimization, incorporates an additional silhouettes-overlap term. Finally, the model's surface is refined by moving the silhouette rim vertices towards the corresponding 2D silhouette contours, via Laplacian deformation.

De Aguiar et al. [46], abandoning the use of a skeleton structure, use a low-resolution tetrahedral mesh for low-frequency tracking, via a volumetric Laplacian deformation toolbox and exploiting silhouette and SIFT features correspondences. The deformation of the tetrahedral mesh is then transferred to its high-resolution triangle mesh counterpart, which is finally updated via standard Laplacian deformation.

The works of Straka and his colleagues [47, 48] have also to be mentioned here. Based mainly on silhouette constraints from multiple cameras, they achieve high-rate performance. More specifically, in the RapidSkin work [47], focusing mainly to the skeleton-free deformation part, they propose a novel fast solver to iteratively solve nonlinear constraints, in order to deform a roughly pose-aligned model using silhouette and Laplacian-deformation smoothness constraints. In [47] they derive a problem formulation that allows to jointly optimize for pose and shape and can be efficiently computed. They propose an alternative to transformation-based skinning methods that finally allows to obtain an optimal deformation using linear solvers.

Most of the works so far require green-screen background (for robust sil-

houette extraction) and several cameras to faithfully track and reconstruct the surface geometry. A recent method [49] however works with a single stereo rig. Based on appearance cues, scene flow and stereo coherence, it achieves foreground extraction. Tracking is realized by minimizing a function that incorporates silhouette, depth (extracted as part of scene flow computation) and shading consistency constraints. The method is slow, requiring computations of several minutes. Still, the achievements are remarkable, given the use of a single stereo pair.

Depth sensor-based methods: In the work of [50], a custom active IR stereo pair is employed, for generating compelling depth input. With highly optimized GPU code, reconstruction of slowly deforming objects is achieved at high frame rates. The method, during a "template acquisition" phase constructs a model of the user, which is then fitted to the live 3D data based on dense model-to-data constraints and as-rigid-as-possible (ARAP) regularization. The method's results are remarkable; however, it has been demonstrated with a single sensor at close distances and no textured results are given. Additionally, as admitted by the authors, the method can handle only a limited amount of frame-to-frame deformation.

The work of [51] proves the concept of using a model-based approach with Kinect cameras for skeleton-based motion capture of human actors. Dou et al. [52] have presented a system for acquiring a human 3D template model by fusing multi-frame data from eight Kinect sensors. The template is then deformed on a per-frame basis to align with the live 3D data, by tracking a set of uniformly sampled nodes. The method is the first one demonstrating non-rigid tracking from a set of commodity RGB-D sensors. However, the

method accounts only for low-frequency deformations and since it does not employ texture mapping, it cannot reproduce facial expressions. Finally, its tracking part requires a computational time of one minute per frame, significantly high compared to the proposed one.

In a more recent promising work [53, 54], Ye et al. achieve fast robust pose and shape estimation from a depth camera, by embedding the articulated deformation model into a probabilistic framework. The method does not require explicitly establishing 3D correspondences between the human template and the observed point cloud. Instead, based on the assumption that the observed point cloud follows a Gaussian Mixture Model (GMM), whose centroids are the vertices of the deformed template, pose estimation is cast as a negative log-likelihood minimization problem, which is solved iteratively using Expectation-Maximization (EM). Their surface estimation algorithm is seamlessly combined with the pose estimation component within the same probabilistic framework. Through an optimized CUDA implementation, the authors achieve a high frame-rate performance.

Recently, Ichim and Tombari [55] proposed another depth camera-based algorithm for pose and shape modeling estimation. The problem is formulated as a global energy minimization problem that includes 3D correspondences (point-to-plane) and contour constrains, combined with 3D feature constraints on the joint positions (obtained as the output of NITE or Kinect2 skeleton tracking), as well as prior energy constraints to limit the tracking algorithm within a learnt/trained subspace and temporal smoothness constraints. It aims to estimate both the skeletal pose and the global shape of the body (as the weighted average of BlendShapes) in two corresponding

alternating optimization stages.

Other, high-quality systems

A complete system for high-quality FVV of humans has been recently described in [2]. An expensive professional studio makes use of a green-screen stage with uniform lighting, encircled by totally 106 synchronized high-speed cameras, 53 HD RGB cameras and 53 IR cameras paired with IR laser sources. The method, combining RGB stereo, IR stereo and SfS, produces a set of separate point-clouds, which are combined using a Moving Least Squares (MLS) projection approach. A visual hull-based extension of Poisson reconstruction [21] is proposed for meshing. To enable efficient mesh compression, key-frames are detected and a non-rigid ICP-based approach is employed for mesh tracking and time-coherent tessellation between key-frames. The whole pipeline performs off-line at rates of 0.5 minutes/frame.

2. Preliminaries and method's overview

2.1. Capturing of multi-view RGB-D videos and TVM reconstruction

While the proposed methodology is applicable with data from any RGB-D sensor, the MS Kinect v2 was adopted in our multi-view capturing platform. As a compromise between system's complexity and coverage area, $K = 4$ sensors are utilized, uniformly arranged on a circle and all pointing to the center of the captured area, as described in our previous work [26].

Time-Varying raw and watertight reconstructions

During the on-line phase of the proposed system, a per-frame TVM reconstruction is needed to guide both articulated pose tracking and surface

deformation (sections 4 and 5). To obtain the required per-frame reconstruction, initially, a simple and fast approach is used to extract the set of "foreground" pixels $\mathbf{u} \in \mathcal{F}_k$ on the k-th depth-map $D_k(\mathbf{u})$. A "raw" 3D point $\mathbf{x}_k(\mathbf{u}) = \Pi_k^{-1}(\mathbf{u}, D_k(\mathbf{u}))$ is then reconstructed for each pixel, where $\mathbf{x} \leftarrow \Pi_k^{-1}(\mathbf{u}, Z)$ describes the projective-to-world mapping of the k-th depth camera. Apart from the "raw" point positions, the corresponding normals $\mathbf{n}_k(\mathbf{u})$ are also estimated. For details, please refer to [26].

Although the raw 3D data might be sufficient, for reasons explained in subsection 4.5 we proceed by reconstructing a watertight surface. This is achieved by applying the volumetric Fourier Transform (FT)-based reconstruction method [21], using our GPU CUDA-based implementation [26], which can reconstruct the human surface at a volume resolution of $128 \times 256 \times 128$voxels in approximately 100msec.

2.2. Performance capture overview

The overall approach is described in the diagrams of Fig. 1 and consists of two phases:

I. **Off-line phase:** Generation of a deformable 3D model, look-alike of the user, via: A) Reconstruction of a smooth watertight 3D mesh, with the user in a relatively "neutral" pose; B) Automatic rigging and skinning of the model, i.e. fitting a pre-defined skeleton and calculation of skinning weights; C) Definition of valid human-pose constraints.

II. **On-line phase:** Per-frame deformation of the human model, through: A) Skeleton-based deformation (implicitly skeleton tracking) of the model to "explain" the captured RGB-D data: This is performed by continuous

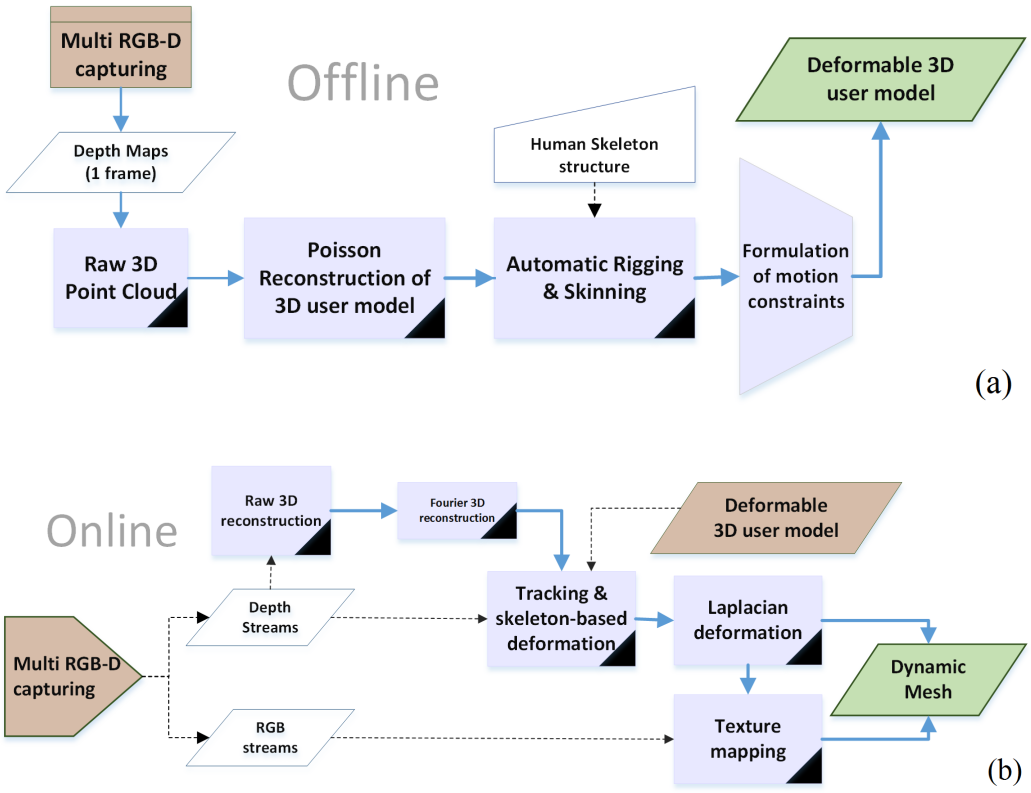
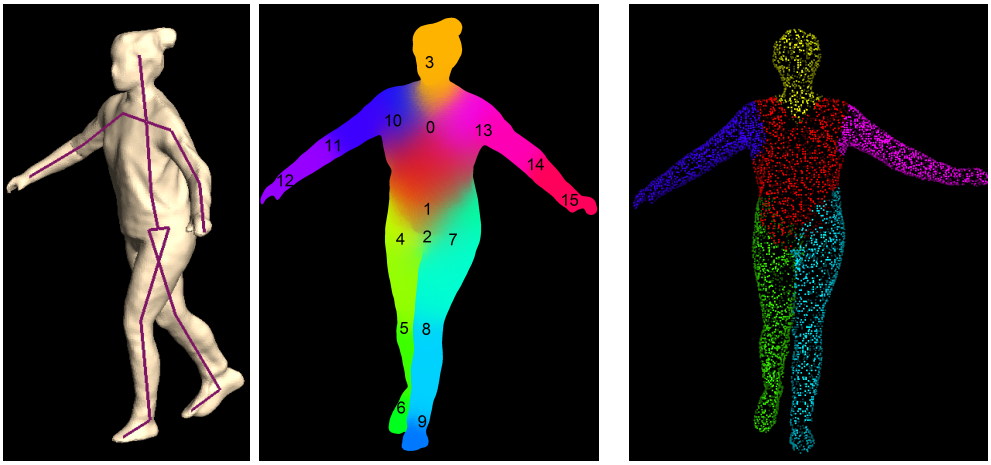


Figure 1: Overview of the proposed method. The method consists of the (a) Offline and the (b) Online phase. See text for details.



(a) Skeleton fitting and skinning weights (b) Handle vertices

Figure 2: (a) Automatic Pinocchio [56] rigging and skinning; (b) The set of handle vertices $\mathcal{H}_0$, encoded with color based on model's segmentation.

"tracking"/optimization, in a three-Levels framework; B) Laplacian deformation of the model, within an iterative framework, to handle highly non-rigid deformations (e.g. garments' wrinkles) and finely "match" the input 3D data; C) Texture mapping.

3. Off-line human template modelling

3.1. Human model reconstruction & automatic rigging

While a 3D user scanning approach could be used, e.g. exploiting a range scanner in a KinectFusion framework [57], as e.g. in the work of [52], we employ a simpler and much faster approach. Since the multi-Kinect capturing platform is available, the user is captured in a "neutral" pose. A raw point-normal cloud is obtained from a single frame, which is then turned into a watertight manifold mesh via Poisson surface reconstruction [20]. Poisson

reconstruction is applied with a tree-depth equal to $r = 8$, generating models of approximately 30000 vertices / 60000 faces.

We note that capturing a strict "neutral" pose, such as a T-pose, is not a prerequisite, as shown in Fig. 2(a). The pose should simply be not too complex, so that automatic skeleton embedding can be realized. The "Pinocchio" method [56] is exploited to realize such a skeleton embedding (rigging) and define how the surface is deformed by the skeletal motion (skin attachment). Focusing on standard linear blend skinning (LBS), Pinocchio assigns vertex-to-bone weights based on the proximity of the embedded bones, smoothed by a diffusion equilibrium equation over the surface. We use the default Pinocchio's skeletal structure, consisting of 18 joints (including the root). An example for the "Lenia" actor, is given at the left of Fig. 2(a). In our application, the two small bones at the soles are not rotated, therefore, ignoring them, our skeletal structure consists of $J = 16$ joints. A color-coded visualization of the skinning weights, is given at the right of Fig. 2(a).

Model's coarse segmentation:

Given the skinning weights, the model is also automatically segmented into 6 district parts, the trunk, the head, the arms and the legs. This segmentation/labeling is performed by finding for each vertex the maximum weight and assigning the corresponding label.

Definition of handle vertices $\mathcal{H}_0$

A subset of the model vertices is additionally selected as a set of "handles". This set, denoted as $\mathcal{H}_0$, is used during the online tracking process, as will be described in subsection 4.3. An example is given in Fig. 2(b). Specifically, the handle vertices are selected randomly, with probability 30%

at the arm limbs, 15% at the legs and the head, 10% at the trunk regions. The denser selection of handles at the arms is intuitively justified by the facts that i) the arms are smaller than other body parts, i.e. have fewer vertices; ii) in most scenarios, their motion is faster and more complicated, and therefore more difficult to be tracked.

3.2. Human pose parameterisation

Rotation formulation using the exponential map

Among the various alternatives available to parameterize 3D rotations of joints (rotation matrices, Euler angles, or quaternions), an axis-angle representation is employed, via the Exponential Map [32, 30, 58]. Specifically, the practical exponential map-based formulation described by Grassia [58] is used: An exponential map of a 3D vector $\boldsymbol{\omega}$ to a unit quaternion $\mathfrak{q} = \{q_s, \mathbf{q}_v\}$ is given from:

$$\mathfrak{q} = e^{\boldsymbol{\omega}} = \begin{cases} \{\cos\left(\frac{\theta}{2}\right), \hat{\boldsymbol{\omega}} \sin\left(\frac{\theta}{2}\right)\}, & \text{if } \boldsymbol{\omega} \neq \mathbf{0} \\ \{1, \mathbf{0}\}, & \text{if } \boldsymbol{\omega} = \mathbf{0}, \end{cases} \tag{1}$$

where $\theta = |\boldsymbol{\omega}|$ and $\hat{\boldsymbol{\omega}} = \boldsymbol{\omega}/|\boldsymbol{\omega}|$ are the angle and axis of rotation, respectively.

To simplify the task of applying constraints to ball-and-socket joints, the rotation can be thought as being composed of a 2-DOF "swing" component, and a 1-DOF "twist" component that defines the bone's rotation around itself. In order to define a 2D basis for the swing rotation, two orthogonal unit vectors are selected, $\hat{\mathbf{s}}_1$ and $\hat{\mathbf{s}}_2$ on the plane perpendicular to the unit bone axis $\hat{\mathbf{b}}$, specifically from: $\hat{\mathbf{s}}_1 = \hat{\mathbf{b}} \times \hat{\mathbf{b}}_\mathrm{p}$ and $\hat{\mathbf{s}}_2 = \hat{\mathbf{s}}_1 \times \hat{\mathbf{b}}$, where $\hat{\mathbf{b}}_\mathrm{p}$ is the direction of the parent bone. A desired swing rotation is given by the exponential map of the vector $\boldsymbol{\omega}_s = p_{s_1}\hat{\mathbf{s}}_1 + p_{s_2}\hat{\mathbf{s}}_2$. The twist rotation is

parameterized by a single angle value, let p_t, about the bone axis. Using the above notation, the parameters vector $\mathbf{p} = [p_{s_1}, p_{s_2}, p_t]$ defines the 3D rotation of a joint.

In our implementation, apart from the root joint that is represented by 6 parameters (translation and 3 Euler angles), all other joints are represented by 3 DOFs, and therefore, the pose is described by a vector $\mathbf{P}$ of length $N_p = 6 + 3(J-1) = 51$, as the concatenation of all joints parameters $\mathbf{p}_j, j = 0, \ldots, J-1$.

We have to highlight that the Exponential Map (in its skew-symmetric matrix formulation [32]) is used in many relevant methods [30, 31, 35, 34, 36]. These however, in contrast to our method, parameterize 3-DOFs joints as the concatenation of 1-DOF (revolute) joints, which introduces limitations in terms of singularities [59].

Human joints rotation limits

Rotation constraints are implicitly applied through the definition of a penalty function, which is zero when rotation remains inside a "valid" range and increases smoothly as the rotation goes outside this range, in order to assist smooth convergence of the online optimization procedure. Specifically, with respect to the swing component, ellipsoidal angular constraints are used. For each joint, an elliptical region $\mathcal{E}(\mathbf{r}, \mathbf{c})$: $(p_{s_1} - c_1)^2/r_1^2 + (p_{s_2} - c_2)^2/r_2^2 \leq 1$, defines the limits for vector $\mathbf{p}_s = [p_{s_1}, p_{s_2}]$. The penalty, outside the ellipse, is given by:

$$f_s(\mathbf{p}) = D^2\Big(\mathbf{p}_s; \mathcal{E}(\mathbf{r}, \mathbf{c})\Big)/C^2, \tag{2}$$

where $D^2\Big(\mathbf{p}_s; \mathcal{E}\Big)$ is the squared Euclidean distance of $\mathbf{p}_s$ from the ellipse and $C = 15^{\text{o}}$ a pre-defined scale factor. With respect to the twist component,

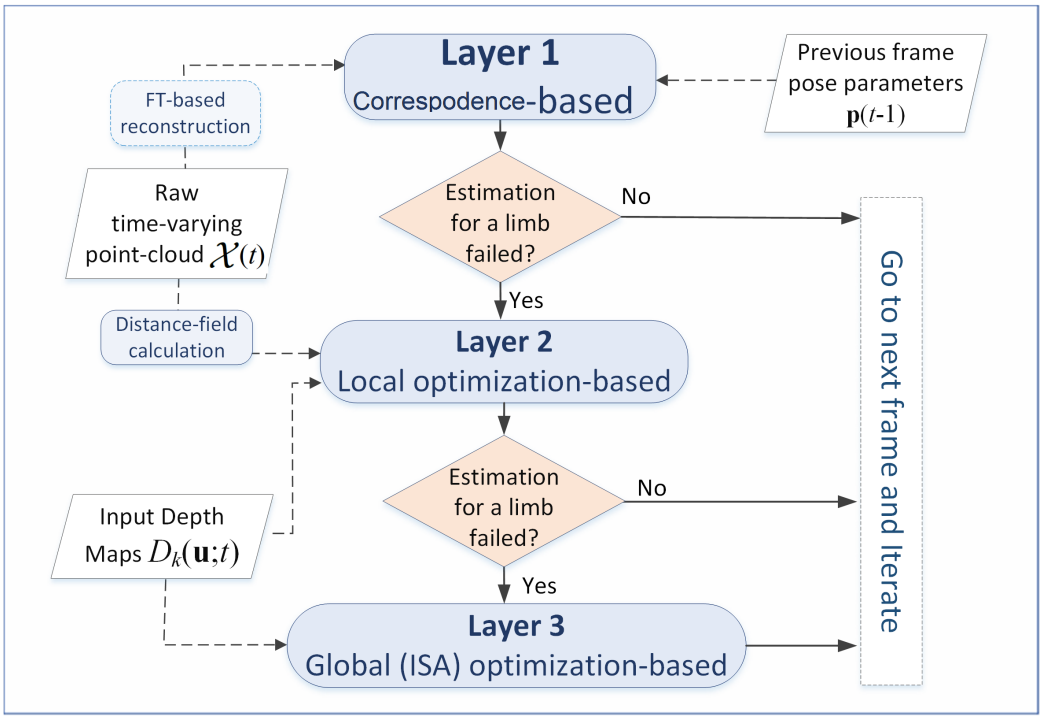


Figure 3: The three-layers skeleton-based pose estimation approach.

a penalty $f_t(p)$ is defined, which equals the squared distance of p_t from a "valid" range $[p_t^{\min}, p_t^{\max}]$, normalized by C^2. The total penalty equals the sum of both, i.e. $f(\mathbf{p}) = f_s(\mathbf{p}) + f_t(\mathbf{p})$.

4. Skeleton-based pose estimation

4.1. A three-layers approach

The proposed method estimates the pose parameters $\mathbf{P}(t)$ for the current frame t, in a tracking framework, given the previous frame's parameters $\mathbf{P}(t-1)$. It consists of three distinct layers, shown in Fig. 3. Layer #1 is executed for each frame, whereas a next layer is executed only if needed, in

the case of a limb's tracking failure. Each next layer offers higher tracking robustness, being however slower.

Given the reconstructed time-varying mesh for the current frame, Layer #1 estimates all $N_p = 51$ pose parameters simultaneously. Iteratively, it finds target positions for the model's handle vertices and estimates the optimum pose parameters that minimize handles' distances to those target positions.

On the other hand, the major task of Layer #2 is to re-enforce a limb towards its correct pose, when its tracking has been lost. Layer #2 operates by minimizing a sophisticated energy function, via a local optimization algorithm. It handles each misaligned limb separately, increasing speed and robustness: Given the fact that the torso is well aligned by Layer #1 (always verified in practice), there is no obvious reason to handle all misaligned limbs jointly, which would increase computational effort and the chance to get trapped in a bad local minimum.

Whereas practically failure of Layer #2 is rare, Layer #3 was added to ensure that no human manual intervention is needed in the whole framework. Handling also each misaligned limb separately, it searches for the global minimum of the same energy function via a slow, but robust global optimization algorithm, the Interacting Simulated Annealing (ISA, [38]).

4.2. A 3D data association approach

Both the pose estimation algorithm of Layer #1 and the surface deformation approach of subsection 5.1 need a 3D data association mechanism, which computes a target 3D position for a vertex of the source model. A mechanism similar to the one of [60] is proposed. For a given source vertex of the template model, the nearest points of the target mesh are initially searched,

within a sphere of radius R. The points with normals that significantly deviate from the normal of the source vertex are rejected. Specifically, the inner product of the source normal and each target point's normal is tested against a threshold $\alpha_{\max}$. From the remaining target points, their weighted centroid is computed, with the weights selected from $w = \frac{R-d}{R}$, where d is the Euclidean distance of a target point from source. Finally, the target displacement is restricted only along the normal direction, to avoid tangential drift.

We highlight that in some cases, the source point may be not associated with a target position, because no target points with "similar" normals exist in its spherical vicinity. This is the case when tracking of a limb has been lost and the model limb's vertices are not close enough to any target raw points.

4.3. Layer #1: Correspondence-based pose estimation

The pose estimation algorithm of Layer #1 can be summarized as follows: Given the pose of the model in the previous time-instance, for each handle vertex of the deformed model $\mathbf{v}_i(\mathbf{P}), i \in \mathcal{H}_0$ the corresponding target positions $\mathbf{x}_i$ are estimated using the method of subsection 4.2. Then, using the Levenberg-Marquardt (LM) non-linear optimization algorithm [61, 62], the "robust" point-plane error

$$E(\mathbf{P}) = \sum_{i \in \mathcal{H}_0} \left[1 + f_{j(i)}(\mathbf{P})\right] \psi\Big(\mathbf{n}_i^{\mathrm{T}}(\mathbf{x}_i - \mathbf{v}_i(\mathbf{P}))\Big) \tag{3}$$

is minimized, where $\psi()$ is the robust Tukey penalty and $f_j(\mathbf{P})$ is the natural-pose constraint function, calculated as described in subsection 3.2. The sub-

script $j(i)$ denotes the joint to which the i-th vertex is mostly associated. $\mathbf{n}_i = \mathbf{n}_i(\mathbf{P})$ stands for the normal of the i-th model's vertex.

Sequentially, the model is deformed using the estimated optimum parameters and the method is iteratively applied in exactly the same way. In each iteration however, the search radius R for the 3D data association method is halved and the inner-product threshold $\alpha_{\max}$ is increased. In this way, having ensured that each previous iteration brings the model closer to the actual pose, the pose "detail" is sequentially improved. Practice showed that three iterations are adequate. In all our experiments, the initial search radius and the inner-product threshold have been set R=300mm and $\alpha_{\max}$=0.5.

We note that the FT-based reconstructed surface (see subsection 2.1) constitutes the input to the proposed algorithm, although it practically works equally well with the raw 3D point-cloud as input. The only reason for selecting the FT-based reconstruction is explained in subsection 4.5.

4.4. Layers #2 and #3

4.4.1. Definition of the energy function

In Layers #2 and #3 the pose parameters are estimated by minimizing a sophisticated energy function. In the ideal case, one would like to minimize a 3D Hausdorff distance-like function, i.e. the distance of the model to the input 3D point-cloud and vice-versa. Such an approach would however require a closest-point search process, which is generally computationally expensive, for each tested parameter vector $\mathbf{P}$. Therefore, it would be prohibitive in our problem, given the nature of Layers #2 and #3, which involve several iterations / parameter tests. Thus, with respect to the model-to-input points distance, we introduce a distance metric that is based on the calculation of a

volumetric distance-field. The calculation of this field is fast and performed only once at each time instance. With respect to the input cloud-to-model distance, we introduce a function that is calculated on the 2D depth image plane and its calculation can be very fast (a few msec). Additionally, a silhouette-based metric is introduced.

3D distance field-based energy

The objective is to calculate a scalar function $F(\boldsymbol{\chi})$, defined over a discrete 3D grid, which describes the Euclidean distance of each voxel $\boldsymbol{\chi}$ from the nearest point in the input cloud. To that end, the human bounding box is discretized into voxels of size 1cm^3. A binary volume function $B(\boldsymbol{\chi}) \in \{0, 1\}$ is initially constructed, by "splatting" each sample $\mathbf{x}_i$ to its containing voxel. The separable algorithm of [63] is then used to compute the exact Euclidean Chamfer distance transformation of $B(\boldsymbol{\chi})$, denoted as $F(\boldsymbol{\chi})$. The selection of the specific algorithm was based on its ability to calculate the exact transformation in a linear time ($\text{O}(N)$, where N is the number of voxels) and due to its separable nature that enabled our efficient GPU implementation. An example distance field is given in Fig. 4.

Given the volumetric distance field, the model-to-input cloud distance energy is calculated from:

$$E_1(\mathbf{P}) = \sqrt{\frac{1}{I}\sum_{i=0}^{I-1}\Big(F\big(\boldsymbol{\chi}[\mathbf{v}_i(\mathbf{P})]\big)\Big)^2}, \tag{4}$$

where $\boldsymbol{\chi}[\mathbf{v}]$ denotes the voxel that contains vertex $\mathbf{v}$, while $\mathbf{v}_i(\mathbf{P})$ is the i-th vertex of the model. Given the definition of the distance field $F(\boldsymbol{\chi})$, the energy $E_1(\mathbf{P})$ equals the RMS closest-point distance of the model from the input point-cloud and is expressed in mm.

Energy metrics on the 2D image plane

The specific metrics are calculated by i) projecting the model mesh onto the K depth-sensor planar views, in order to obtain the depth maps $D_k^{\mathrm{M}}(\mathbf{u};\mathbf{P})$ and ii) "comparing" the "rendered" depth with the input foreground-segmented depth maps $D_k(\mathbf{u})$. From an implementation point-of-view, using the OpenGL-CUDA interoperability framework, computations can be extremely fast. The energy metrics described below, are also explained in Fig. 5.

Closest point-like energy: For each view $k = 0, \ldots, K-1$, the depth map $D_k^{\mathrm{M}}(\mathbf{u};\mathbf{P})$ is back-projected onto the 3D space to generate the 3D points $\mathbf{v}_k(\mathbf{u};\mathbf{P})$. Similarly, we have the raw 3D points $\mathbf{x}_k(\mathbf{u})$ from the input depth map. For each foreground pixel $\mathbf{u} \in \mathcal{F}_k$ of the k-th input depth map, we define the truncated squared distance metric:

$$e_k(\mathbf{u};\mathbf{P}) = \min\Big(\min_{\mathbf{u}_1 \in \mathcal{N}(\mathbf{u})} \big\{||\mathbf{x}_k(\mathbf{u}) - \mathbf{v}_k(\mathbf{u}_1;\mathbf{P})||^2\big\}, T^2\Big), \tag{5}$$

where $\mathcal{N}(\mathbf{u})$ is a rectangular neighbourhood around $\mathbf{u}$, of size 31×31 pixels and T=150mm is a truncation distance threshold, selected based on the size of $\mathcal{N}(\mathbf{u})$ and the camera's intrinsic parameters. The overall energy is given from:

$$E_2(\mathbf{P}) = \sqrt{\frac{1}{\sum_k |\mathcal{F}_k|} \sum_k \sum_{\mathbf{u} \in \mathcal{F}_k} e_k(\mathbf{u};\mathbf{P})}, \tag{6}$$

where $|\mathcal{F}_k|$ denotes the number of foreground pixels in the k-th view. The energy E_2, like E_1, is also expressed in mm.

Silhouette-based energy: Additionally, the binary 0/1 silhouette masks $S_k(\mathbf{u})$ and $S_k^{\mathrm{M}}(\mathbf{u};\mathbf{P})$ are obtained from the corresponding depth maps. A silhouette-similarity energy metric is defined from:

$$E_3(\mathbf{P}) = \frac{50}{\sum_k |\mathcal{F}_k|} \sum_k |S_k \oplus S_k^{\mathrm{M}}(\mathbf{P})|, \tag{7}$$

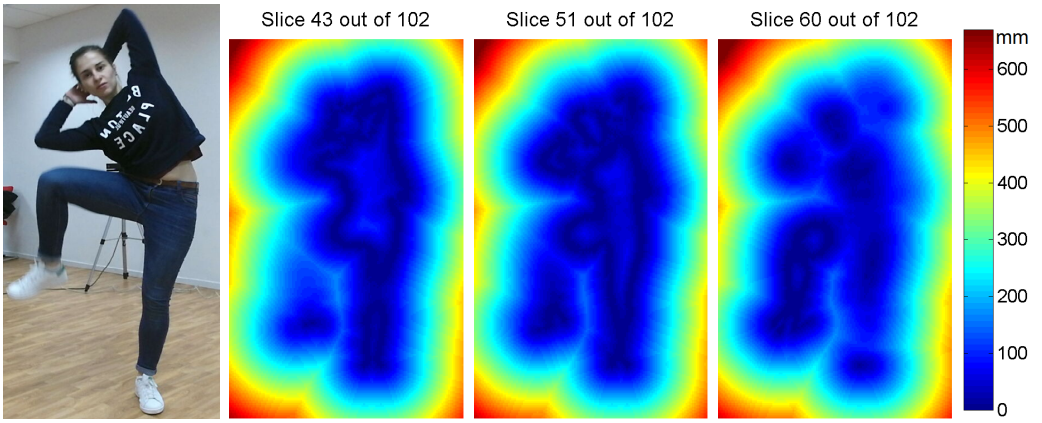


Figure 4: Volumetric Euclidean distance field. Three slices Z=const of the distance field are shown, along with a corresponding RGB view.

where $\oplus$ denotes the binary XOR operator and $|\cdot|$ counts for the number of true (1) elements. The normalization factor $\frac{50}{\sum_k |\mathcal{F}_k|}$ is used to intuitively assign to every non-matched (w.r.t. silhouette) foreground pixel a distance of 50mm.

Collision-to-trunk penalty

To ensure avoidance of undesirable situations where a limb penetrates the trunk, an additional penalty term is introduced. Instead of using a standard collision-detection algorithm, which would be computationally demanding and would result into a binary 0/1 discontinuous function, the notion of Truncated Signed Distance Function (TSDF) [18, 64] is used. Given that TSDF is calculated from a set of input depth images, we project the trunk segment of the model onto a set of 8 virtual (OpenGL) cameras, equally distributed on a circle of radius 2000mm. This is performed only once, at the beginning of Layer #2. Given a truncation threshold μ, the TSD equals $-\mu$ inside the object, μ outside the object and varies smoothly from μ to $-\mu$

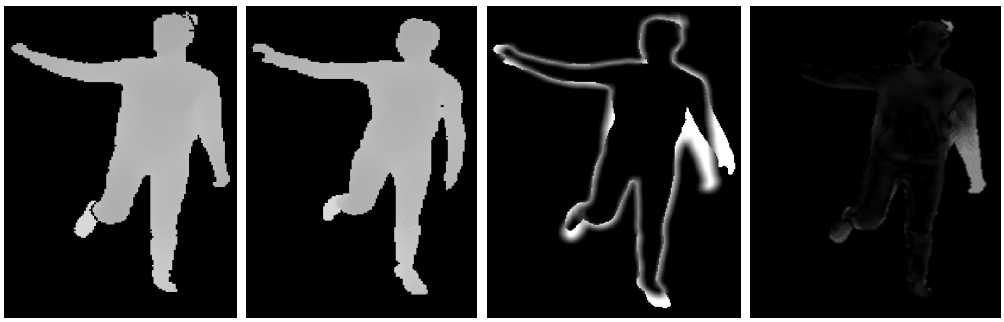


Figure 5: From left to right: i) Captured depth map; ii) Depth of the model; iii) Per-pixel silhouette energy; iv) Per-pixel, closest-point distance energy.

near the object's surface. In our algorithm, the TSDs of the arm vertices are calculated. The minimum TSD is considered as the distance of the arm to the trunk. Let this minimum TSD be denoted as $d(\mathbf{P})$. A collision-to-trunk penalty is defined from $f_c(\mathbf{P}) = \frac{1-d(\mathbf{P})/\mu}{2} \in [0, 1]$. In our experiments, μ was set equal to 6cm.

Combined energy function

The combined energy to be minimized is given by:

$$E(\mathbf{P}) = \big[1 + f_c(\mathbf{P})\big] \cdot \prod_{j=1}^{J-1}(1 + f_j(\mathbf{P})) \cdot \sum_{n=1}^{3} w_n E_n(\mathbf{P}), \tag{8}$$

where $E_n(\mathbf{P}), n = 1, 2, 3$ are given from (4), (6) and (7), respectively, and w_n are weights, set equal to unity in our experiments. The factor $\prod(1 + f_j(\mathbf{P}))$ is used to express the natural-pose constraints, defined in subsection 3.2.

4.4.2. Layer #2: Downhill simplex optimization

Even by handling each limb separately, the dimensionality of the parameters space is quite high; e.g. the pose of an arm is affected by three joints (shoulder, elbow, wrist), i.e. 9 parameters. Therefore, a fast, approximate,

gradient-free optimization algorithm is used. Specifically, the Nelder-Mead downhill simplex method [65, Ch. 10] is selected, which is expected to require fewer function evaluations per iteration than gradient-based methods.

4.4.3. Layer #3: Global ISA-based optimization

Layer #3 seeks for the globally optimal solution using ISA, a stochastic, particle-based, global optimization approach. The ISA algorithm iterates between a weighting, a selection and a mutation step, evolving the particles' population towards the optimum solution. For details, the reader is referred to [38, 36]. Here, only the necessary details for its application in our framework are given: In our experiments, a) The number of ISA iterations is fixed to 8 and the number of particles to M=2000; b) The initial particles' population $\mathbf{P}_m, m = 1, \ldots, M$, is uniformly sampled in the ranges that express the human joints' rotation constraints (see subsection 3.2); c) The temperature-like parameter β_i, used in the weighting step, decreases with iterations i according to $\beta_i = (1+i)^{0.7}$; d) The mutation step is implemented similarly to [36], with the difference that we ensure that each "mutated" particle lies within the ranges of natural-pose joints' constraints.

4.5. Additional issues

Tracking initialization

For pose initialization in the first frame, Layer #3 is employed. Specifically, assuming that the actor in the 1st frame stands in a relatively neutral (not extreme) pose, the initialization of each body segment is performed sequentially: Firstly, the torso position-orientation (6 DOFs) is initialized, and the limbs' poses are then separately estimated. With respect to torso

initialization, two issues have to be highlighted: A) The energy functions in equations (4)-(7) are calculated considering only the vertices of the torso segment; B) Assuming that the user looks towards the frontal camera, the ISA particles are initialized and constrained inside the interval $[-15^{o}, 15^{o}]$, for each Euler angle. With respect to position, the centroid of the raw input mesh is calculated and the population is initialized to be $\pm$300mm around this centroid, along each direction. Five ISA iterations are applied with $M = 1000$ particles. We note that accurate initialization is not of great importance, because the pose parameters converge to more accurate values during the first frames of the sequence, where Layer #1 is applied.

Automatic detection of limb tracking failure

It was practically found that a simple heuristic, which is based on the employed 3D data association algorithm (subsection 4.2), can serve for the robust detection of loss of tracking: For each limb, the percentage of model's handle vertices without associated target positions, denoted as $C\%$, is computed. The data association algorithm is applied with $R = 75$mm and $\alpha_{\max} = 0.7$, i.e. with the same parameters used in the last iteration of Layer #1. This means that we practically check the percentage $C\%$ during the last iteration of Layer #1. When the percentage $C\%$ is above a threshold T, the limb is considered as mis-detected and the next pose-estimation Layer is applied. In our experiments, the used threshold is $T = 15\%$ when checking the output of Layer #1 and $T = 40\%$ at the output of Layer #2. According to our experiments, these values are adequately small to ensure that loss of tracking is always detected. Although several false "alarms" may be generated, this is not an important issue, as explained in the experimental

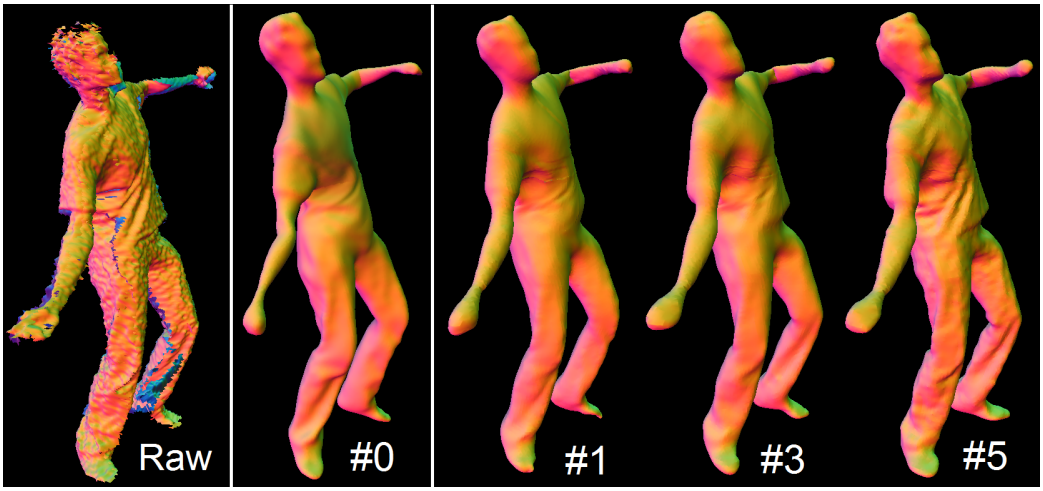


Figure 6: Example of progressive shape refinement, based on iterative Laplacian deformation. From left-to-right: i) Raw 3D data, ii) Initial model (iteration #0), iii) Refined model at iterations #1, #3 and #5. The color encodes the surface normals (no back-face culling is applied).

section.

As foretold, the proposed algorithm works equally well when using as input the raw 3D data or the FT-based reconstructed mesh. However, the FT-based reconstruction is used as the target surface, due to the following reason: Self-occlusions result into non-captured/reconstructed areas in the raw 3D point-cloud. In contrast, the FT-based reconstruction produces a watertight surface, regardless of occlusions, resulting practically into more stable behaviour of $C\%$ during tracking. Consequently, the selection of appropriate thresholds is easier and the described heuristic method works more effectively.

5. Shape refinement and texture mapping

The skeleton-based deformation can lead to only a coarse match of the model to the captured 3D data, since it is based on the assumption that the model's deformation is explained only in terms of the underlying skeleton. Fine-scale surface detail is missing from the deformed surface. Thus, the shape has to be refined, i.e. its vertices have to be smoothly shifted, to better match the input 3D data. To this end, the coupling of model vertices to the skeleton structure is abandoned and an iterative deformation framework is proposed.

5.1. Laplacian deformation within an iterative framework

Similarly to other relevant works [43, 46, 35], we use a Laplacian deformation framework [44, 45]. In contrast to these works however, which are based on the human's contour/rims 2D information that may be difficult to be extracted and may not be adequate for complicated shapes with many concavities, we use directly the available input 3D information. The idea is based on the iterative refinement of the model's shape, guided by local 3D correspondences that are established using the 3D data association algorithm of subsection 4.2.

The iterative algorithm can be summarized as follows: In each iteration, the target positions $\{\mathbf{x}_i\}$ of handle model vertices $\{\mathbf{v}_i\}, i \in \mathcal{H}(s)$, are computed using the data association algorithm of subsection 4.2. In the first iteration, only each s_0-th vertex of the model ($s_0 = 8$ in our experiments) is considered as a handle vertex, and the search radius for the association algorithm is set to $R = 150$. In each next iteration, the number of handles

is doubled ($s \leftarrow 2 \cdot s$), until all vertices are used as handles in the last iterations, i.e. $s_{\mathrm{e}} = 1$, and the search radius is halved ($R \leftarrow R/2$). In this way, each next iteration accounts for finer details. Practice showed that five iterations are adequate. With respect to the inner-product threshold $\alpha_{\max}$ of the association algorithm, it is fixed to $\alpha_{\max} = 0.7$ ($\sim 45^{\mathrm{o}}$) in our experiments.

Given the target positions (handle constraints) $\mathbf{X} = [\mathbf{x}_0^{\mathsf{T}}, \mathbf{x}_1^{\mathsf{T}}, \ldots]^{\mathsf{T}}$ in each iteration, the energy

$$E(\mathbf{V}) = ||\mathrm{C}\mathbf{V} - \mathbf{X}||^2 + ||\mathrm{L}\mathbf{V} - \boldsymbol{\delta}||^2 \tag{9}$$

is minimized to compute the model's new vertex positions $\mathbf{V} = [\mathbf{v}_0^{\mathsf{T}}, \mathbf{v}_1^{\mathsf{T}}, \ldots]^{\mathsf{T}}$, i.e. to find a smooth deformation of the model. Here, L is the cotangent Laplacian matrix, $\boldsymbol{\delta}$ are the differential coordinates, and C is a diagonal matrix with non-zero weights only for constrained (handle) vertices, all set to unity in our implementation. The second term in (9), which uses the Laplacian, is a regularization term for the deformation defined by our constraints. The minimization of (9) is a linear least-squares problem that is solved using direct sparse LDLT Cholesky factorization.

An example is given in Fig. 6. As can be seen, the initial model (skeleton-based only deformed) is aligned with the raw point-cloud, but its high-frequency components (e.g. garment wrinkles) do not match the real ones. Employing the proposed deformation method, the shape of the model is iteratively refined.

5.2. Texture mapping

A significant task for creating FVV video, which is not addressed by most of the relevant "performance capture" methods, is the mapping of texture

to the generated meshes. In contrast to TVM reconstruction methods, e.g. [26], the proposed framework can take advantage of the fact that the generated sequence is dynamic and there is one-to-one vertex correspondence from frame to frame. Thus, in our method the model's texture at a time instance is obtained using both the live RGB frame and previous frames. Before going into details, we initially describe a novel RGB pre-processing algorithm that can enhance the final visual quality.

Handling background-on-foreground artifacts

In consumer-grade RGB-D devices, the RGB and depth sensors are not synchronized and the calibration (mapping) between them is not perfect. Thus, texture artifacts may appear, where the color of the background is mapped along the boundaries of the reconstructed mesh. To partially overcome such problems, a pre-processing step is proposed, applied to each RGB view separately. The method consists of the following steps: a) Given the silhouette map on the RGB image plane, apply two iterations of Close morphological operation, using a circular mask of radius equal to 2pixels; b) Apply two iterations of Erosion filtering, with a circular mask of radius equal to 3pixels, to remove $L{=}2{\times}3{=}6$ layers of silhouette's boundary pixels (i.e. remove 6 times the silhouette contour); c) Augment the foreground pixels into the background region, by assigning to each adjacent background pixel the mean RGB value of the foreground pixels in its 3×3 neighborhood. The latter step is applied iteratively, $2L - 1$ times. The sizes of the masks were selected based on the resolution of the Kinect2 RGB images (1920×1080). The objective of (a) is to close small holes (e.g. due to missing data in Kinect depth views), so that they are not eroded by the next step (b). Steps (b)

and (c) aim at replacing the wrongly assigned color of the background along the boundaries of the human silhouette. The reader can find an explanatory example in the supplementary material document. The number of iterations and mask sizes were selected intuitively and experimentally; similar morphological operator settings could be used to achieve the described objectives.

Combining RGB views at a single time instance

In order to produce the color of a vertex from the available RGB views at a specific time instance, a color weighting approach [26] is employed. Instead of using equal weights for all RGB views, i) the "viewing" angle of the captured surface, as well as ii) the closeness of the 2D projection pixel to the human boundaries on the color image plane, are taken into account. For details, the reader is referred to [26]. For simplicity in what follows, let us denote as $\mathfrak{T}(\mathbf{v}_i, \mathcal{I})$ the process of calculating the color of vertex $\mathbf{v}_i$ from the set of live RGB views $\mathcal{I}$.

Handling invisible vertices

To handle vertices that are invisible to all RGB cameras at a specific time instance, the colors found in previous frames are used. If $c_i(t)$ denotes the color of vertex $\mathbf{v}_i$ at time t, this is updated from frame to frame using:

$$c_i(t) = \begin{cases} c_i(t-1), & \text{if } \mathbf{v}_i \text{ invisible to all cameras,} \\ \mathfrak{T}\big(\mathbf{v}_i, \mathcal{I}(t)\big), & \text{otherwise,} \end{cases}$$

This way, we maintain a color-per-vertex representation of the model's texture.

Applying UV-texture mapping

A color-per-vertex rendering approach may lead to color aliasing, resulting into low visual quality, due to the fact that the resolution of the model

Figure 7: Texturing process. From left to right: i) An original RGB view; ii) Deformed model, with color-per-vertex using the RGB information only in the current frame. Gray vertices are those that are invisible to all cameras; iii) Final rendered model, using color-per-vertex for invisible vertices and UV-texture mapping for the visible ones.

mesh is not in general very high, compared to the resolution of the original RGB views. Therefore, we employ UV-texture mapping, where each visible triangle is assigned multiple texture patches from the live RGB views and rendered with OpenGL multi-texture blending, using the associated vertex weights. Only triangles that are invisible to all cameras are rendered using the color-per-vertex model's representation. An example is given in Fig. 7.

Real-time FVV rendering

The output textured meshes, although generated off-line, can be straight-

forwardly rendered in real-time frame-rates with the use of OpenGL shaders. With our shaders, the generated meshes can be rendered at frame-rates >30fps.

6. Experimental Results

The test data were recorded with the multi-Kinect capturing setup described in subsection 2.1 and comprise of 10 sequences, with 4 different actors, 3 males ("Apostolakis", "Alexandros" and "Dalexiad") in 4 sequences and one female ("Lenia") in 6 sequences, performing a set of various motions that range from simple to challenging ones, such as kick-boxing actions or fast athletic movements. In total, the sequences consist of more than 16000 frames, i.e. we have approximately 11 minutes of action at ∼25fps. The RGB-D data, along with the necessary software for parsing, can be found at `http://vcl.iti.gr/performancecapture/dataset2/`. Apart from the results presented here, a supporting-material document, along with a large number of long videos, can be found at `http://vcl.iti.gr/performancecapture/`, demonstrating that the proposed algorithm can faithfully track and reconstruct a wide spectrum of actions.

6.1. Implementation details and processing hardware

For all the implementation details, please refer to our supporting-material document. We note here that the 3D data association algorithm (subsection 4.2) and the calculation of the energy metrics of subsection 4.4.1, were parallelized to run on the GPU using CUDA. The rest of the code, including all optimization methods (LM, Downhill-simplex and ISA), run on the CPU,

Figure 8: Apostolakis2 sequence - Jumping jacks. Left: Two consecutive frames; Right: The corresponding raw 3D data, the tracked skeleton and the corresponding "puppeted" model, for the 2nd frame.

specifically on a single CPU thread. Thus, we see that room for implementation improvement exists and code optimization/parallelization can lead to further run-time reduction. The presented experiments ran on a PC with an i7 processor (3.4GHz), 16GB RAM, a CUDA-enabled NVidia GTX560 GPU and a Windows7 operating system.

6.2. Qualitative evaluation

6.2.1. Tracking

As can be verified from Figures 8, 9 and the relevant Figures in the supplementary document, as well as from the supporting-material videos, the proposed method can robustly track a wide range of challenging motions, including "tennis", "punching and kicking", "jumping", "jumping jacks", etc. In Fig. 8, the raw 3D data coming from different Kinect sensors are not perfectly aligned (especially at the arms), due to the very fast motion (indicated also by the motion blur in the original views) and the imperfect synchronization between Kinect sensors. Even under such imperfect synchronization con-

Figure 9: Frames from the six Lenia sequences and the corresponding tracked skeletons. The sequences are encoded as "Lenia1",..., "Lenia6".

ditions, the method works efficiently. The proposed method, combining all three layers, did not lose tracking with the used sequences (>16000frames), and no human manual intervention was actually needed.

The diagrams in Figures 10, and 11 depict the percentage $C\%$ of non-associated vertices of the human arms (see subsections 4.2 and 4.5) at the output of Layer #1, along time. With the results in these diagrams, we aim at partially demonstrating the tracking efficiency of Layer #1, as well as the appropriateness of the selected threshold 15%. In Fig. 10(upper row), where a left/right hand punching sequence is studied, the periodicity of the motion is reflected in the percentage $C\%$. Although $C\%$ increases during the fast punching periods, it remains in low levels, below the threshold of 15%, and indeed the arms were well tracked. The diagram in the bottom row of the Figure corresponds to $1\frac{1}{2}$ period of a "jumping jacks" actions. As

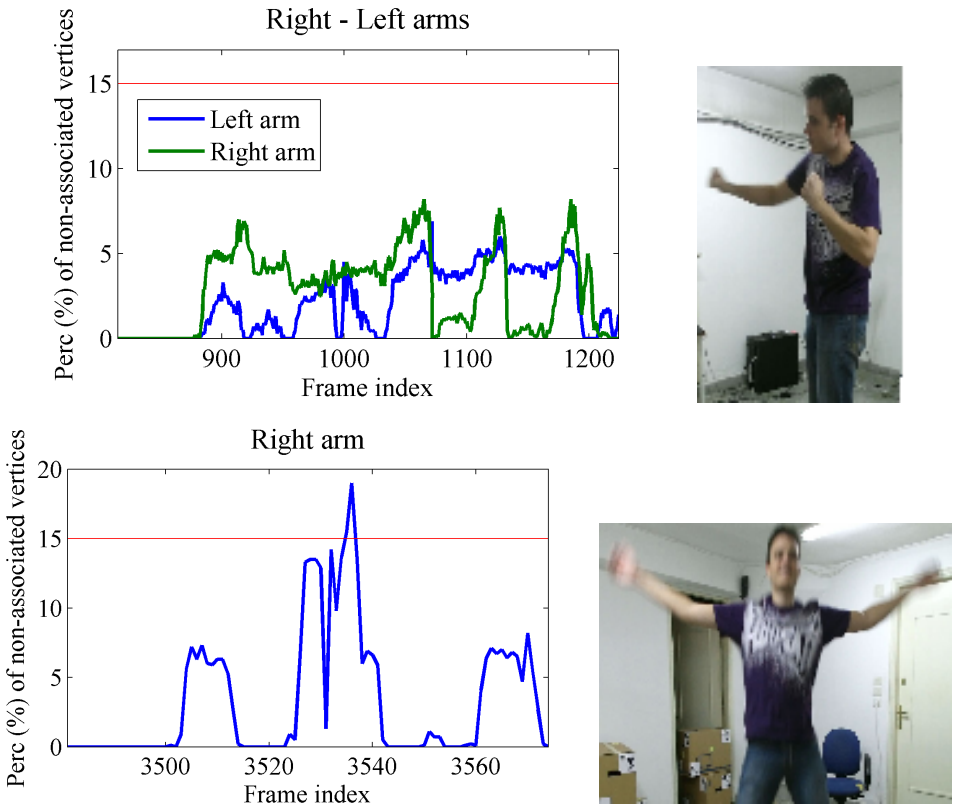


Figure 10: Apostolakis - The percentage of non associated vertices at the output of Layer #1. Only during the very fast "jumping jacks" case, the threshold of 15% is reached. See text for details.

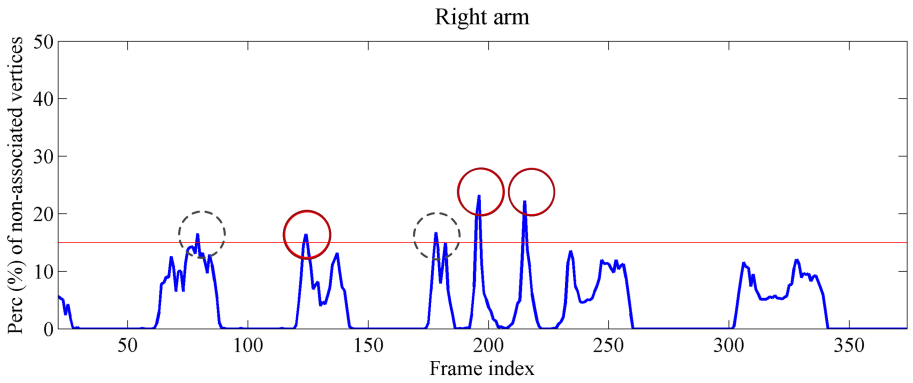


Figure 11: Lenia3: Detection of Layer #1 tracking failure. The red and the dashed gray circles indicate true and false positives, respectively.

can be seen, $C\%$ becomes larger than the threshold of 15% at a single time instance, where indeed tracking was lost by Layer #1. In this case, Layer #2 re-enforced the arm to the correct pose.

It was practically observed that the most challenging case for the proposed method is the one where the user keeps or moves slowly her/his arm(s) completely stacked on the body trunk. The diagram of Fig. 11 corresponds to "Lenia" performing periodically an exercise. The periodicity is reflected in the diagram. Between two exercise periods, she keeps (or moves slowly) her whole arms stacked on her trunk, for short intervals. At these short intervals, the percentage $C\%$ of non-associated vertices presents peaks/spikes, higher than 15%, resulting into "alarms" and executions of Layer #2. These "alarms" have been manually annotated as true or false positives. As can be seen, approximately half of them are false. However, they are few, introducing only a minor computational overhead due to the execution of Layer #2.

In total, considering all sequences, Layer #2 was executed in less than 3%

Table 1: Layer #1 tracking time (msec), for the male sequences

	Dalexiad	Apostolakis1	Apostolakis2	Alexandros	All
Processed frames	2921	3272	2639	2216	11048
Average time (msec)	925	1075	1101	827	**992**

of the frames (452 in total). Layer #3 and ISA-based optimization, apart from the body-pose initialization (subsection 4.5), was executed only four times. Thus, Layer #3 proved to be useful mainly for overriding the need of any human manual intervention.

Computational times: Given the implementation details of subsection 6.1, Table 1 provides the mean execution time of Layer #1, for the male sequences. As can be seen, this computational time is slightly smaller than 1sec/frame. Although it does not result into real-time operation, it is significantly lower than the tracking time of similar approaches, which may range from a few seconds [29, 43, 35] to several seconds or minutes [52, 46, 49, 36]. With respect to Layer #2, the mean operation time calculated over all Layer #2 executions, is as low as 1.29sec/limb. Finally, the mean time for Layer #3 is 143sec/limb. Although Layer #3 is slow, its limited usage in our method is not prohibitive.

6.2.2. Surface estimation

An example with respect to the iterative Laplacian deformation of the model was given in Fig. 6. An additional example is given in Fig. 12. As can be verified, the final output surface is watertight, smooth and noise-free, while it contains most of the geometric details of the actual data, such as the clothes garments. Several additional results can be found in the supplementary-

material document and videos, at `http://vcl.iti.gr/performancecapture/`.

Apart from the major advantage of producing dynamic mesh sequences, the proposed model-based method has the advantage of being robust to self-occlusions, as demonstrated in Fig. 13. This figure depicts the raw 3D data for a specific pose, the corresponding "puppeted" model and the final deformed surface. Comparing the final output at Fig. 13(d) against the per-frame model-free Poisson reconstruction (volume resolution 128^3 voxels) at Fig. 13(b), it can be verified that the proposed method can efficiently handle self-occlusion and concavities, due to its model-based nature.

Finally, the method is more robust to "ghost limb" problems, compared to per-frame model-free methods, as demonstrated in Fig. 14: Due to fast motion and imperfect synchronization of the sensors, the separate input raw data are not perfectly aligned, as shown in Fig. 14(a), especially at the fast moving hands. This results into problematic reconstruction of any model-free per-frame volumetric method, as the Poisson method in Fig. 14(b), in contrast to the proposed one.

Computational times: Apart from the 3D data association algorithm, which was implemented on the GPU, the other elements of iterative Laplacial deformation algorithm run on a single CPU thread. The mean computation time, calculated over all sequences' frames, is 2.52 sec/frame, including the calculation of the Laplacian and the 5 iterations of deformation. Although higher than the corresponding time for tracking, the computation time for our surface refinement step remains in quite low levels, compared to reported times of relevant methods, e.g. in similar levels with [35] and an order of magnitude lower than in [46, 49].

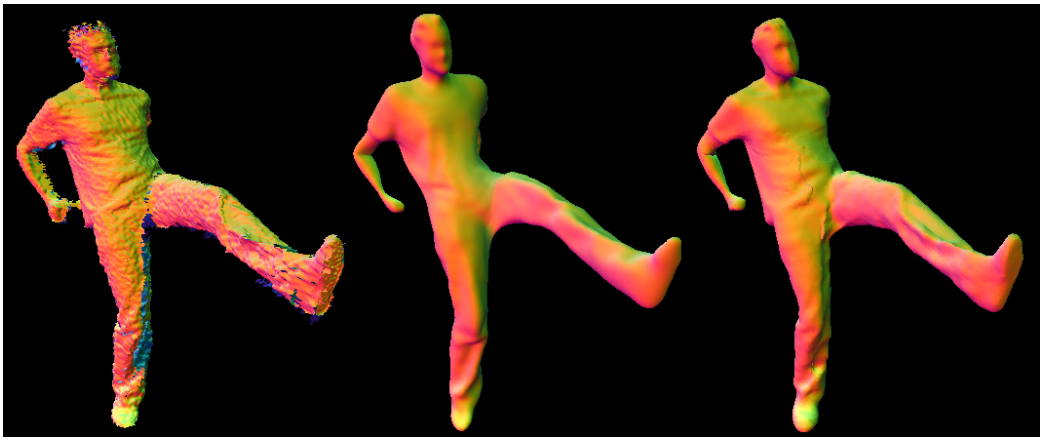


Figure 12: From left to right: Raw data, "puppeted" and deformed model.

Figure 13: Lenia6: (a) Raw input 3D data; (b) Poisson surface reconstruction (volume resolution $128 \times 128 \times 128$ voxels); (c) "Puppeted" model; (d) Output surface of the proposed model-based method. Check the concavity between the right arm and the head.

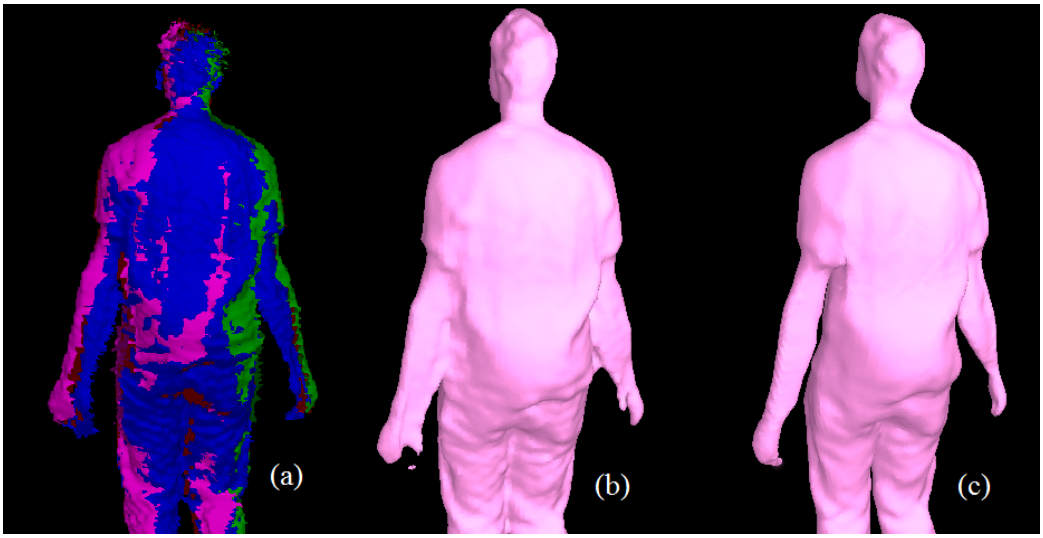


Figure 14: Apostolakis: (a) Raw, separate meshes encoded with different colors; (b) Poisson reconstruction ($128 \times 128 \times 128$); (c) Proposed method.

6.2.3. Qualitative comparison vs model-free methods

We also compare against a VolumeDeform-like [28] model-free method that dynamically fuses depth information from consecutive frames via ARAP [66] deformation-based tracking and volumetric TSDF fusion [18]. Specifically, we compare against our multi-view implementation of VolumeDeform, where (compared to [28]) i) the information from multiple RGB-D cameras (four in our case) is used; ii) For dense depth correspondences finding, the projective data association of [28], which according to our experiments does not work for large motions, has been replaced with our 3D data association of subsection 4.2; c) AKAZE texture features [67] are used, in place of the patented SIFT features. According to our experimental results, although a VolumeDeform-like framework produces high quality results in the case of relatively slow motions and with the human being captured at close

Figure 15: Alexandros - From left-to-right: Reconstruction by (a) a VolumeDeform-like [28] approach at frame $t-5$ and (b) at frame t; (c) by the proposed model-based at frame t. Large motion cannot be efficiently handled by a model-free method, even with the use of RGB-based 2D features, as in [28].

distances, such a method is not as robust as a model-based one in challenging human motions, like those used in our experimental results. There are cases where large motions cannot be handled well, probably due to the fact that the generic ARAP deformation model is not restrictive enough for such challenging motions. An example is given in Fig. 15. The accumulation of new depth information after inaccurate tracking seems to produce destructive results, as shown in both Figs. 15 and 16. As a future work direction, we would like to investigate how one could combine a model-based method, such as the proposed one, with a VolumeDeform-like model-free method, to take advantage of both.

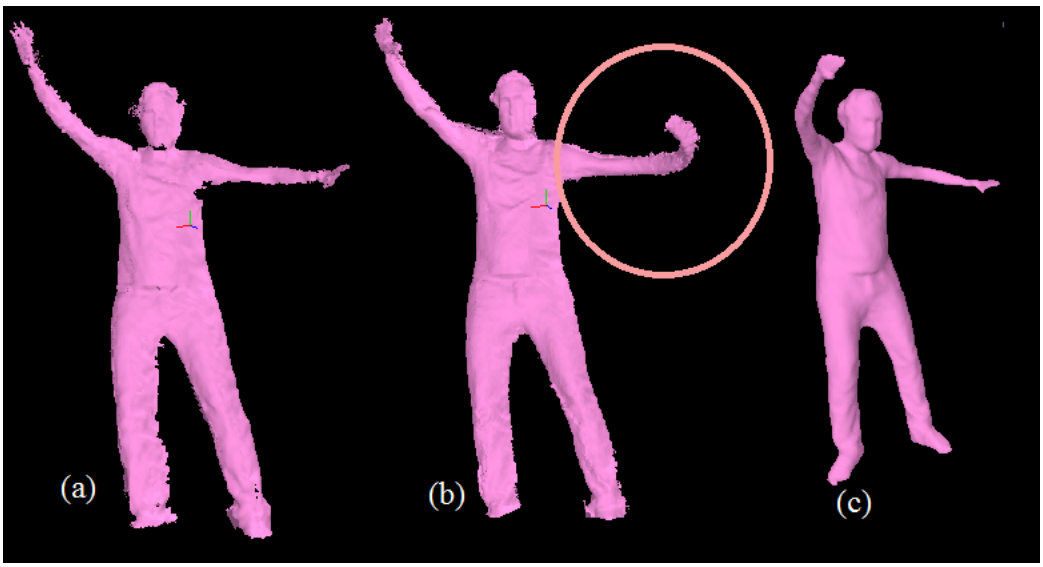


Figure 16: Alexandros - From left-to-right: (a) Raw reconstruction; (b) VolumeDeform-like reconstruction [28]; (c) proposed model-based reconstruction. The proposed model-based reconstruction is less sensitive to tracking inaccuracies, compared to model-free methods.

Figure 17: Lenia4 - From left-to-right: (a) Raw textured data; (b) Model-based, color-per-vertex; (c) Model-based with UV texture-mapping for the visible areas. Bottom row: Zoom in the face.

6.2.4. Texture mapping

Apart from Fig. 7, textured reconstruction results are given in Fig. 17 and the supplementary-material document and videos. The textured results demonstrate that the deformed surface is well aligned with the input 3D data; otherwise, texture-misalignment artifacts would be observed.

At the left of Fig. 17, the raw textured reconstruction is depicted, without the application of the input RGB pre-processing step (see subsection 5.2). The "background - on - foreground" artifact is visible: The white background is mapped along the human's silhouette. Additionally, the raw reconstruction is noisy and of low-quality. On the other hand, the textured (color-per-vertex) deformed model, shown at the middle, presents less artifacts. Additionally, the whole mesh is textured, despite the fact that some mesh regions are invisible to all RGB views of the current frame. However, the low resolution of the mesh results into a "blurred" output view. The finally rendered view, using the proposed weighted UV-texture mapping approach, is given at the right. The texture is sharper and no significant visible artifact exists.

6.3. Quantitative evaluation

6.3.1. Evaluation sequences with known Ground-Truth

To quantitatively evaluate the proposed methodology and compare it against existing methods, multi-view depth sequences were generated by projecting an animated 3D model onto virtual depth cameras. Specifically, the "Dalexiad", "Lenia" and "Apostolakis" skinned models were animated using the skeleton data tracked by our method for the "Apostolakis1", "Apostotakis2" and "Dalexiad" sequences, respectively. The corresponding 3D

Figure 18: An example frame of the evaluation data. Left: Ground-truth mesh; Middle: Depth maps generated by projecting onto the frontal and rear virtual cameras; Right: The depth maps with AWGN and holes.

mesh sequences were projected onto four virtual depth cameras, with intrinsic and extrinsic parameters exactly the same with the actual Kinect sensors. The generated sequences, with known geometry and skeleton Ground-Truth (GT) are referred to as "VDalexiad", "VLenia" and "VApostolakis" and can be found at `http://vcl.iti.gr/performancecapture/dataset2/`, along with the necessary details. An example is given in Figure 18. Either all four views or only a subset of them is used in our experiments, as explained in the following paragraph.

6.3.2. Experiments and quantitative results

The proposed method is evaluated with respect to both pose estimation (skeleton tracking) and surface geometry accuracy. The method was executed using i) all four depth views, ii) only the frontal and rear views, as well as the 2 views with iii) Additive White Gaussian Noise (AWGN) to the depth of standard deviation equal to 20mm and iv) the same AWGN plus randomly missing depth data (holes) of density 30%. An example is given at the right of Fig. 18. We highlight that when applied with only two views, the raw

3D data are used as input (two views are not adequate to produce a well-define watertight mesh via FT-based reconstruction) and the corresponding thresholds for tracking-failure detection (subsection 4.5) are set to $T = 25\%$ and $T = 55\%$ for the 1st and 2nd layer respectively, to avoid several "false alarms".

With respect to the skeleton tracking (pose estimation) results, the Root Mean Squared (RMS) error against the joint positions GT is used as the evaluation metric. With respect to surface geometry estimation, the Metro tool [68] (`https://sourceforge.net/projects/vcg/`) is used to calculate the RMS distance between the estimated triangular mesh and the GT mesh, and vice versa, which are denoted as E-GT and GT-E, respectively.

Pose tracking

The method is compared against the "offline" model-based pose tracking method of [55], using the code provided by the authors at `https://github.com/aichim/bodies-ras-2015`. This method takes as input a depth map sequence, as well as a noisy skeleton sequence (normally obtained as the output of a skeleton tracker, such as NITE or Kinect2 skeleton module) and simultaneously tracks the pose and estimates the human shape as a weighted average of blendshapes. Since NITE or Kinect2 skeleton data are not available, the GT skeleton is used itself as the noisy skeleton sequence input. As for the weights of its various cost terms, the default parameters of their implementation are used, apart from the weight of the "skeleton feature constraints" terms, which was set to 0.9 times the default value, since the use of the GT skeleton favours significantly the method.

Figure 19 gives example comparative results, illustrating the per-frame

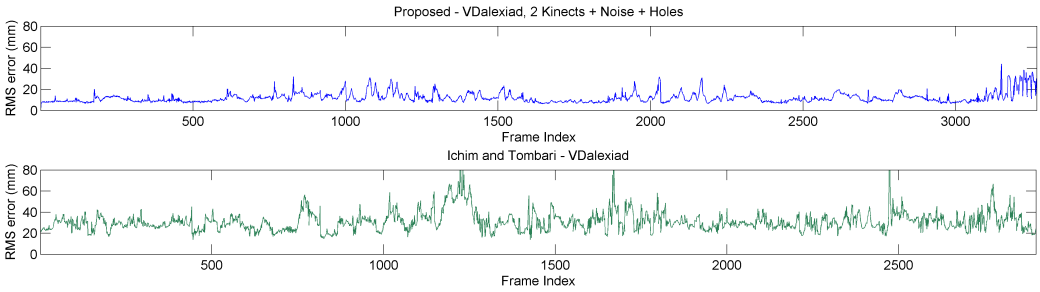


Figure 19: Per-frame RMS error for the "VDalexiad" sequence. Upper row: Results of the proposed method (2 views with AWGN and missing data); Bottom row: Results of [55].

RMSE of the proposed method vs the method of [55]. Similar results for other sequences are given in the supplementary material document. According to our experimental evidence, [55] works quite robustly with our data. However, it is not as accurate as the proposed method, although it is favoured by taking as input ("skeleton feature constraints") the GT skeleton data. The reason is probably the fact that [55] does not use a custom model for a specific actor. Instead, it aims to estimate the global shape of the user body, as the weighted average of generic blendshapes and cannot handle well loose clothing. The all-frames results for all sequences are summarized in Table 2, showing that the accuracy of the proposed method significantly outperforms [55].

Surface estimation results

With respect to the reconstructed geometry, the method is compared against a model-free approach, a multi-view implementation of VolumeDeform [28], as explained in paragraph 6.2.3. Additionally, since VolumeDeform [28] is based on TSDF fusion [18], the results of a per-frame multi-view TSDF approach are also given.

Since the tracking part of the model-free approach fails at the relatively

Table 2: Quantitative evaluation and comparative results: RMS Error (mm) of the tracked joint positions with respect to the GT positions.

Method and input	"VDalexiad"	"VLenia"	"VApostolakis"
Proposed, 4 views	8.45	9.39	3.95
Proposed, 2 views	9.67	11.09	5.31
" , " + AWGN	11.83	13.22	9.87
" , " + AWGN + Holes	12.09	14.77	11.03
Ichim & Tombari [55]	30.73	38.09	30.86

fast motion parts, producing destructive results (see paragraph 6.2.3), in order to have a more fair comparison, the method's output is "reset" whenever tracking is lost (based on manually supervising the output at each frame). This means that the output volume is reset to the TSDF function for the specific frame and the tracking-fusion method continues normally to the next frames.

The RMS Distances GT-E and E-GT for the first 450 frames of "VDalexiad" with 2 views are given in Fig. 20. The GT-E metric reflects partially the completeness of the reconstructed surface, while E-GT reflects the accuracy of the reconstructed part. As can be verified from the presented diagrams, the GT-E metric for the VolumeDeform-like approach (blue line, upper diagram) starts decreasing after each "reset", until after some frames the method loses tracking and the error starts increasing. The meshes illustrated at the top of upper diagram, demonstrate that a) At frame 178 tracking has been lost; b) Thus, at frame 179 the reconstruction is "reset"; c) After some frames (frame 210) the VolumeDeform mesh is more complete

Table 3: Quantitative evaluation and comparative results: RMS Distance (mm) of the extracted triangular mesh vs the GT mesh (E-GT) and vice-versa (GT-E), as well as their average (AVG).

		"VDalexiad"			"VApostolakis"		
Input	Method	E-GT	GT-E	AVG	E-GT	GT-E	AVG
4 views	Per-Frame TSDF	5.66	10.9	8.29	5.41	10.27	7.84
	VolDeform-like	6.10	8.42	7.62	5.52	8.28	6.90
	Proposed	2.06	2.35	2.21	1.94	2.04	1.99
2 views	Per-Frame TSDF	4.61	20.4	12.1	4.55	19.82	12.19
	VolDeform-like	5.11	14.7	9.91	4.80	13.85	9.33
	Proposed	2.75	2.96	2.85	2.52	2.71	2.61
2 views + AWGN	Proposed	4.02	4.61	4.31	3.72	4.19	3.95
2 views + AWGN+Holes	Proposed	4.16	5.37	4.77	4.03	4.84	4.43

Table 4: Quantitative evaluation of Layer #2 vs Layer #1 and impact of energy terms.

	Tracking loss percentage		
Input	Mode1	Mode2	Mode3
"VDalexiad", 2 views +AWGN+Holes	1.79%	0.52%	1.21%

than the corresponding TSDF mesh.

The corresponding per-frame results for the proposed method (with 2 depth views, with AWGN and missing data) are given in the diagram of Fig. 21. Similar diagrams are given in the Supplementary material document. Table 3 summarizes comparative results, which show that the model-free approach cannot be as robust and accurate as the model-based one.

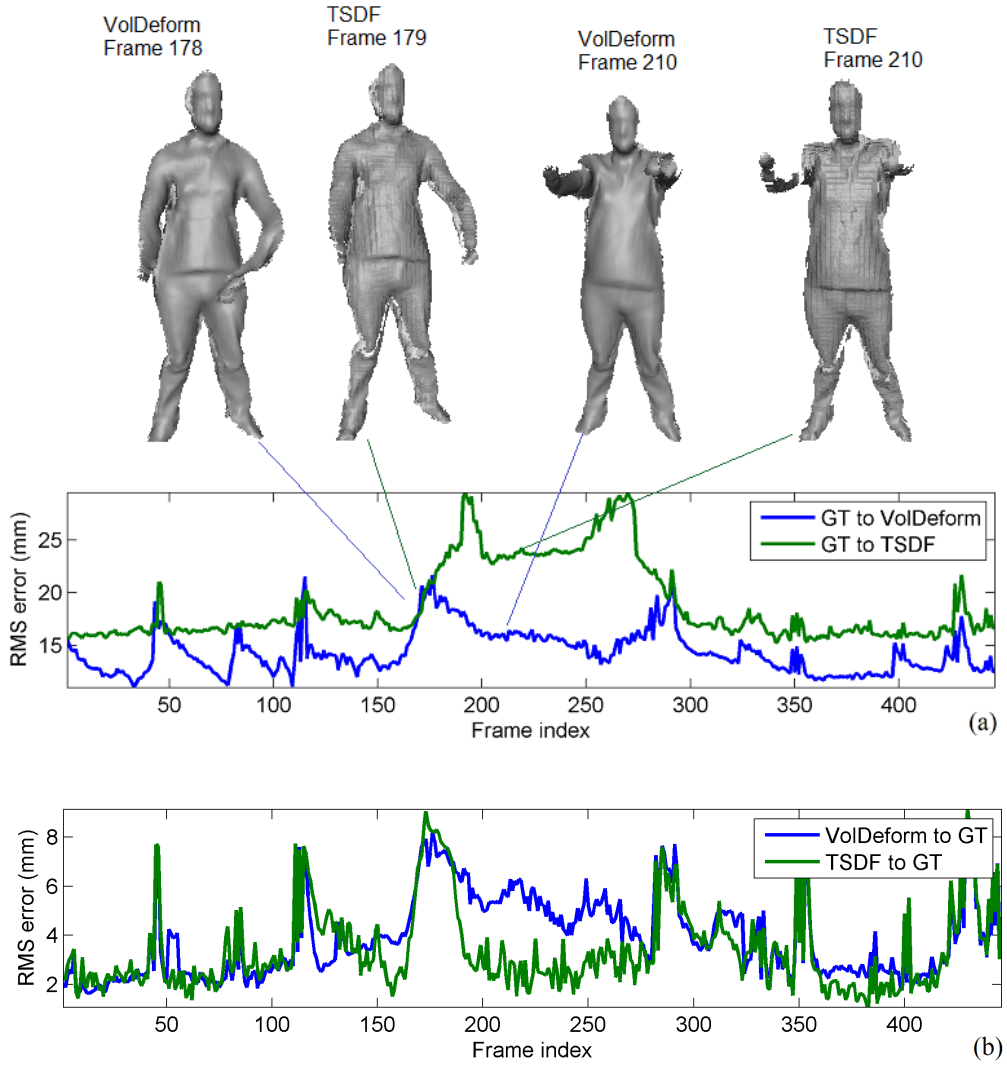


Figure 20: Per frame quantitative results for the VolumeDeform-like approach and the per-frame TSDF, for "VDalexiad" with **2 views**: RMS Distance (mm) of the (a) extracted mesh vs GT (E-GT) and (b) vice-versa (GT-E). Some corresponding VolumeDeform-like and per-frame TSDF meshes are also depicted. See text for details.

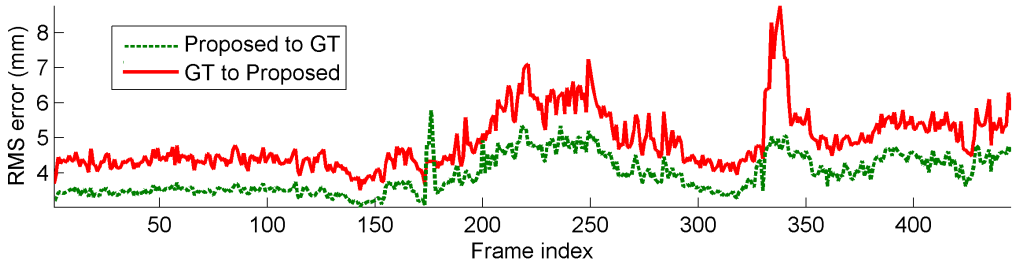


Figure 21: Per-frame quantitative results of the proposed method, for "VDalexiad" with **2 views, AWGN and holes**: The RMS Distance (mm) of the extracted mesh vs GT (E-GT) and vice-versa (GT-E), are given.

6.3.3. Evaluation of Layers and different energy terms

In this section, we are interested in a) demonstrating the efficiency of Layer #2 vs Layer #1, and b) evaluating the importance of the various energy terms of Layer #2. With the term "efficiency", we refer to "robustness" instead of "accuracy", i.e. we are interested in the number of times tracking is lost (note that the mission of Layer #2 is to reinforce mis-tracked joints close to the correct positions).

Towards the above objectives, the algorithm was run in the following modes. i) Mode1: The algorithm runs only Layer #1; ii) Mode2: Layer #1 runs to update the pose, then the pose parameters of the limbs are set to those of the previous frame, and finally Layer #2 runs for each limb (and in each frame); iii) Mode3: The same as Mode3, with the difference that only the energy term $E_1(\mathbf{P})$ is used (see (4)), i.e. the weights are set $w_2 = w_3 = 0$ in (8). In all three modes, whenever tracking is lost, the algorithm continues from the next frame and the pose parameters are set to the ground-truth of that frame. Since the ground-truth is available, loss of tracking is considered

when the joint position error (for any joint) becomes >20cm.

The percentages of frames where tracking is lost in the three different modes are given in Table 4. As can be verified, Layer #2 (Mode2) is more robust than Layer #1 (Mode1), verifying our arguments of using Layer #2 on top of Layer #1, as Layer #1 is based on the establishment of 3D correspondences, which may fail in the case of large motion.

Additionally, the results with Mode3 vs Mode2 demonstrate the necessity of the energy terms E_2 and E_3 (see subsection 4.4.1). The energy E_1 "attracts" the model towards the input 3D mesh. It is a model-to-input point cloud "attractive" energy. However, as stated in subsection 4.4.1 an energy term that accounts for the input point cloud-to-model distance is also needed, to ensure that situations where e.g. the arms remain stacked on the body, are avoided. This argument is verified by the results in Table 4.

6.4. Method's limitations - Discussion

The proposed method shares the limitations of model-based approaches in handling topological and scene changes. On the other hand, we believe that the results of the proposed method, with respect to both tracking and the final output (textured) surface are remarkable, especially when considering the low computation time needed, the relatively low number of the input RGB-D sensors and the inherent sensors' limitations. Still, the output is not always free of artifacts and the visual quality is not perfect when compared to methods such as [2], applied with expensive professional setups of hundreds of perfectly synchronized cameras. For example, in our case, the imperfect synchronization of the RGB streams (both with each other and with the depth cameras) introduces texture artifacts during fast motions. One addi-

tional inherent limitation of the method, due to its model-based nature and the imperfectly synchronized noisy input, is that the method may sometimes not reconstruct faithfully thin structures, such as the hands (fist and fingers). Due to similar reasons and the low-resolution of the user's model, the facial details are smoothed. Towards a solution of the latter, we expect that using models of high resolution at the face will improve the reconstruction of facial details.

7. Conclusions-Future Work

We have presented a novel end-to-end system that recovers the pose and surface motion of an actor, fully automatically, using the input from a few consumer-grade RGB-D sensors. The proposed method, focusing on speed and robustness through a set of novel approaches, makes automatic processing of large data sets feasible. The computational burden due to the high dimensional search space is reduced by the introduction of a layered approach, where each next layer is slower but executed only when needed, as well as by the definition of sophisticated energy functions, fast computable on the GPU. The experimental results on a large number of performances demonstrate the efficiency of the proposed approach and its ability to produce quite realistic FVV.

The limitations of the system have also been explained. Apart from the exploitation of hardware- synchronized capturing equipment, we believe that the use of mesh models of adaptive resolution can help towards the further improvement of visual quality at semantically important regions, such as the face or hands. Additionally, although the proposed method exceeds

significantly the speed performance of related methods, it operates "quick-post". Thus, we envision a real-time realization of the system, suitable for live tele-immersion applications, via the full parallelization of faster optimization methods versions, on high-performance GPUs.

Acknowledgement

This work was supported and received funding from the European Union Horizon 2020 Framework Programme under Grant Agreement no. 761934 (Hyper360).

References

[1] A. Smolic, 3D Video and Free Viewpoint Video - From capture to display, Pattern Recognition 44 (9) (2011) 1958–1968.

[2] A. Collet, M. Chuang, P. Sweeney, D. Gillett, D. Evseev, D. Calabrese, H. Hoppe, A. Kirk, S. Sullivan, High-quality streamable free-viewpoint video, ACM Trans. Graph. 34 (4).

[3] K. Mamou, T. B. Zaharia, F. J. Preteux, FAMC: the MPEG-4 standard for animated mesh compression, in: Proc. ICIP, 2008.

[4] A. Doumanoglou, D. Alexiadis, D. Zarpalas, P. Daras, Towards real-time and efficient compression of human time-varying-meshes, IEEE Trans. Circuits Syst. Video Technol. 24.

[5] G. Vogiatzis, C. Hernandez, P. Torr, R. Cipolla, Multiview stereo via volumetric graph-cuts and occlusion robust photo-consistency, IEEE Trans. Pattern Anal Mach. Intell. 29 (12) (2007) 2241–2246.

[6] Y. Furukawa, J. Ponce, Carved visual hulls for image-based modeling, Int. J. Comput. Vis. 81 (2009) 53–67.

[7] T. Matsuyama, X. Wu, T. Takai, T. Wada, Real-time dynamic 3D object shape reconstruction and high-fidelity texture mapping for 3-d video, IEEE Trans. Circuits Syst. Video Technol. 14.

[8] J.-S. Franco and E. Boyer, Efficient polyhedral modeling from silhouettes, IEEE Trans. Pattern Anal Mach. Intell. 31 (3) (2009) 414 – 427.

[9] G. Haro, Shape from silhouette consensus, Pattern Recognition (9) (2012) 3231–3244.

[10] K. Kutulakos, S. Seitz, A theory of shape by space carving, Int. J. Comput. Vis. 38(3) (2000) 199 – 218.

[11] G. G. Slabaugh, W. B. Culbertson, T. Malzbender, M. R. Stevens, R. W. Schafer, Methods for volumetric reconstruction of visual scenes, Int. J. Comput. Vis. 57(3) (2004) 179 – 199.

[12] S. Prakasha, A. Robles-Kelly, A semi-supervised approach to space carving, Pattern Recognition 43 (2) (2010) 506–518.

[13] T. Kanade, P. Rander, P. J. Narayaran, Virtualized reality: Constructing virtual worlds from real scenes, IEEE Multimedia 4 (1).

[14] C. L. Zitnick, S. B. K. M. Uyttendaele, S. Winder, R. Szeliski, High-quality video view interpolation using a layered representation, ACM Trans. Graph. 23 (3).

[15] R. Vasudevan, G. Kurillo, E. Lobaton, T. Bernardin, O. Kreylos, R. Bajcsy, K. Nahrstedt, High quality visualization for geographically distributed 3D teleimmersive applications, IEEE Trans. Multimedia 13 (3).

[16] L. Kang, L. Wu, Y.-H. Yang, Robust multi-view L2 triangulation via optimal inlier selection and 3D structure refinement, Pattern Recognition 47 (9) (2014) 2974–2992.

[17] G. Turk, M. Levoy, Zippered polygon meshes from range images, in: SIGGRAPH '94, 1994, pp. 311–318.

[18] B. Curless, M. Levoy, A volumetric method for building complex models from range images, in: SIGGRAPH '96, 1996.

[19] H. Zhou, Y. Liu, Accurate integration of multi-view range images using k-means clustering, Pattern Recognition 41 (1) (2008) 152–175.

[20] M. Kazhdan, M. Bolitho, H. Hoppe, Poisson surface reconstruction, in: Proc. 4th Eurographics symp. on Geom. processing, 2006.

[21] M. Kazhdan, Reconstruction of solid models from oriented point sets, in: Proc. 3rd Eurographics symp. on Geom.processing, 2005.

[22] N. Ahmed, C. Theobalt, C. Rossl, S. Thrun, H.-P. Seidel, Dense correspondence finding for parametrization-free animation reconstruction from video, in: IEEE CVPR, 2008.

[23] A.Dipanda, S.Woo, Towards a real-time 3D shape reconstruction using a structured light system, Pattern Recognition 38 (10) (2005) 1632–1650.

[24] A. Maimone, H. Fuchs, Real-time volumetric 3D capture of room-sized scenes for telepresence, in: IEEE 3DTV-Conf., 2012.

[25] D. Alexiadis, D. Zarpalas, P. Daras, Real-time, full 3-D reconstruction of moving foreground objects from multiple consumer depth cameras, IEEE Trans. Multimedia 15 (2013) 339–358.

[26] D. Alexiadis, A. Chatzitofis, N. Zioulis, O. Zoidi, G. Louizis, D. Zarpalas, P. Daras, An integrated platform for live 3D human reconstruction and motion capturing, IEEE Trans. Circuits Syst. Video Technol.

[27] R. Newcombe, D. Fox, S. Seitz, DynamicFusion: Reconstruction and tracking of non-rigid scenes in real-time, in: CVPR, 2015.

[28] M. Innmann, M. Zollhöfer, M. Nießner, C. Theobalt, M. Stamminger, Volumedeform: Real-time volumetric non-rigid reconstruction, in: CVPR, 2016.

[29] J. Carranza, C. Theobalt, M. A. Magnor, H.-P. Seidel, Free-viewpoint video of human actors, ACM Trans. Graph. 22 (3).

[30] G. Pons-Moll, B. Rosenhahn, Visual Analysis of Humans-Looking at People, Ch. 9 Model-Based Pose Estimation, Springer, 2011.

[31] C. Bregler, J. Malik, K. Pullen, Twist based acquisition and tracking of animal and human kinematics, Int. J. Comput. Vis. 56 (3) (2004) 179–194.

[32] R. M. Murray, Z. Li, S. S. Sastry, Mathematical Introduction to Robotic Manipulation, CRC Press, 1994.

[33] T. Brox, B. Rosenhahn, J. Gall, D. Cremers, Combined region and motion-based 3d tracking of rigid and articulated objects, IEEE Trans. Pattern Anal. Mach. Intell. 32 (2010) 402–415.

[34] S. Corazza, L. Muendermann, E. Gambaretto, G. Ferrigno, T. P. Andriacchi, Markerless motion capture through visual hull, articulated ICP and subject specific model generation, Int. J. Comput. Vis. 87 (1) (2010) 156–169.

[35] J. Gall, C. Stoll, E. de Aguiar, C. Theobalt, B. Rosenhahn, H. P. Seidel, Motion capture using joint skeleton tracking and surface estimation, in: IEEE CVPR, 2009.

[36] J. Gall, B. Rosenhahn, T. Brox, H.-P. Seidel, Optimization and filtering for human motion capture: A multi-layer framework, Int. J. Comput. Vis. 87 (1) (2010) 75–92.

[37] J. Gall, B. Rosenhahn, H. Seidel, Clustered stochastic optimization for object recognition and pose estimation, DAGM. Lecture Notes in Computer Science 4713 (2007) 32–41.

[38] J. Gall, J. Potthoff, C. Schnorr, B. Rosenhahn, H. Seidel, Interacting and annealing particle filters:mathematics and a recipe for applications, J. Math. Imaging Vis. 28 (2007) 1–18.

[39] P. Kaliamoorthi, R. Kakarala, Parametric annealing: A stochastic search method for human pose tracking, Pattern Recognition 46 (5) (2013) 1501–1510.

[40] H.-D. Yang, S.-W. Lee, Reconstruction of 3D human body pose from stereo image sequences based on top-down learning, Pattern Recognition 40 (11) (2007) 3120–3131.

[41] X. Zhao, Y. Liu, Generative tracking of 3d human motion by hierarchical annealed genetic algorithm, Pattern Recognition 41 (8) (2008) 2470–2483.

[42] J. Darby, B. Li, N. Costen, Tracking human pose with multiple activity models, Pattern Recognition 43 (9) (2010) 3042–3058.

[43] D. Vlasic, I. Baran, W. Matusik, J. Popovic, Articulated mesh animation from multi-view silhouettes, ACM Trans. Graph.

[44] O. Sorkine, Differential representations for mesh processing, Computer Graphics Forum 25 (4) (2006) 789–807.

[45] M. Botsch, O. Sorkine, On linear variational surface deformation methods, IEEE Trans. Vis. Comput. Graphics 14.

[46] E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmedand, H.-P. Seidel, S. Thrun, Performance capture from sparse multi-view video, ACM Trans. Graph. 27 (3).

[47] M. Straka, S. Hauswiesner, M. Ruether, H. Bischof, Rapid Skin: Estimating the 3D human pose and shape in real-time, in: 2nd International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission (3DimPVT), 2012.

[48] M. Straka, S. Hauswiesner, M. Ruether, H. Bischof, Simultaneous shape and pose adaption of articulated models using linear optimization, in: European Conference on Computer Vision (ECCV), 2012.

[49] C. Wu, C. Stoll, L. Valgaerts, C. Theobalt, On-set performance capture of multiple actors with a stereo camera, ACM Trans. Graph. 32.

[50] M. Zollhöfer, M. Nießner, S. Izad, C. Rhemann, C. Zach, M. Fisher, C. Wu, A. Fitzgibbon, C. Loop, C. Theobalt, M. Stamminger, Real-time non-rigid reconstruction using an RGB-D camera, ACM Trans. Graph. (2014) 33 (4).

[51] G. Ye, Y. Liu, N. Hasler, X. Ji, Q. Dai, C. Theobalt, Performance capture of interacting characters with handheld kinects, in: ECCV, 2012.

[52] M. Dou, J.-M. Frahm, H. Fuchs, Scanning and tracking dynamic objects with commodity depth cameras, in: IEEE Int. Symp. on Mixed and Augmented Reality (ISMAR), 2013.

[53] M. Ye, R. Yang, Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera, in: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014.

[54] M. Ye, Y. Shen, C. Du, Z. Pan, R. Yang, Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera, IEEE Trans. Pattern Anal Mach. Intell. 38 (08) (2016) 1517–1532.

[55] A. E. Ichim, F. Tombari, Semantic parametric body shape estimation from noisy depth sequences, Robotics and Autonomous Systems 75 (2016) 539 – 549.

[56] I. Baran, J. Popovic, Automatic rigging and animation of 3d characters, ACM Trans. Graph. 26 (3).

[57] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. K. andJamie Shotton, S. Hodges, A. W. Fitzgibbon, KinectFusion: Real-time dense surface mapping and tracking, in: Proc. 10th IEEE ISMAR, 2011, pp. 127–136.

[58] S. Grassia, Practical parameterization of rotations using the exponential map, J. Graph. Tools 3 (1998) 29–48.

[59] G. Pons-Moll, B. Rosenhahn, Ball joints for marker-less human motion capture, in: IEEE Workshop Applications of Comp. Vision, 2009.

[60] C. Stoll, Z. Karni, C. Roessl, H. Yamauchi, H.-P. Seidel, Template deformation for point cloud fitting, in: IEEE VGTC conference on Point-Based Graphics, Eurographics Association, 2006, pp. 27–35.

[61] D. Marquardt, An algorithm for least-squares estimation of nonlinear parameters, SIAM J. Appl. Math. 11.

[62] M. Lourakis, levmar: Levenberg-marquardt nonlinear least squares algorithms in C/C++, accessed: Mar. 2016 (July 2004).
URL `http://www.ics.forth.gr/~lourakis/levmar/`

[63] D. Coeurjolly, A. Montanvert, Optimal separable algorithms to compute the reverse euclidean distance transformation and discrete medial axis in arbitrary dimension, IEEE Trans. Pattern Anal Mach. Intell. 29 (3) (2007) 437–448.

[64] D. Alexiadis, D. Zarpalas, P. Daras, Real-time, realistic full-body 3D reconstruction and texture mapping from multiple kinects, in: Proc. IEEE IVMSP, 2013.

[65] W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery, Numerical Recipes in C: The Art of Scientific Computing, 2nd Edition, Cambridge University Press, 1988-1992.

[66] O. Sorkine, M. Alexa, As-rigid-as-possible surface modeling, Eurographics Symposium on Geometry Processing.

[67] P. F. Alcantarilla, J. Nuevo, A. Bartoli, Fast explicit diffusion for accelerated features in nonlinear scale spaces, in: In British Machine Vision Conference (BMVC), 2013.

[68] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, Computer Graphics Forum 17 (2) (1998) 167–174.