Contents lists available at ScienceDirect

Computers & Graphics

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

Technical Section

Real-time virtual fitting with body measurement and motion smoothing☆

Umut Gültepe, Uğur Güdükbay*

Bilkent University, Department of Computer Engineering, Bilkent, 06800 Ankara, Turkey

ARTICLE INFO

Article history:
Received 6 February 2014
Received in revised form
2 June 2014
Accepted 2 June 2014
Available online 17 June 2014

Keywords:
Virtual fitting room
Virtual try-on
Garment fitting
Depth sensor
Motion capture

ABSTRACT

We present a novel virtual fitting room framework using a depth sensor, which provides a realistic fitting experience with customized motion filters, size adjustments and physical simulation. The proposed scaling method adjusts the avatar and determines a standardized apparel size according to the user's measurements, prepares the collision mesh and the physics simulation, with a total of 1 s preprocessing time. The real-time motion filters prevent unnatural artifacts due to the noise from depth sensor or self-occluded body parts. We apply bone splitting to realistically render the body parts near the joints. All components are integrated efficiently to keep the frame rate higher than previous works while not sacrificing realism.

1. Introduction

One of the most time-consuming stages of apparel shopping is trying the apparel on, which is not even possible in online stores. With advances in augmented reality technologies, virtual fitting rooms are slowly taking their place in both real and virtual stores [1,2] to improve the quality of apparel trial experience while making it faster. Advanced virtual fitting rooms show the apparel items either on the video of the user or on a virtual avatar, both scaled to reflect the user's body characteristics [3]. Some studies employ automatic body and garment segmentation [4] and physics-based garment and sewing simulation techniques [5,6] for a better fitting experience.

We present a novel virtual fitting room framework that provides all the basic features expected from such an application, along with enhancements in various aspects for higher realism. These enhancements include motion filtering, customized user scaling, and the use of a physics engine. The motion filtering process starts with temporal averaging of joint positions in order to overcome the high noise of the depth sensor. However, temporal averaging does not prove to be sufficient because unnatural movements take place due to limited recognition capabilities and self-occlusion. We implement customized joint angle filters, along with bone splitting, to let limbs twist in a more natural way. We also employ filtering on hip and knee joints to overcome the footskating problem.

The cloth pieces to be fitted on the user's avatar must first be scaled accordingly. To this end, we implemented body measurement process, which starts with depth map smoothing, in order to reduce noise. Afterwards, we utilize the filtered depth map along with filtered user joints to measure a set of parameters, which are used in conjunction to estimate the body height and shoulder width. These parameters are averaged over time to minimize the error.

The physics engine utilizes collision spheres and capsules to perform collision detection. We determine the correct sphere radii and positions during body measurements. The virtual avatar is aligned with a set of invisible spheres and capsules that are aligned with joints and limbs, which are updated in real time and used in collision detection. Cloth particles are also affected by gravity and inertia.

The rest of the paper is organized as follows. First, we give related work on virtual fitting rooms and depth sensors. Next, we provide an overview of the proposed approach, as well as the details of the cloth simulation engine, depth map filtering, body measurements, temporal averaging, and techniques used to cope with the inferior motion data problems. Then, we provide experimental results. Finally, we give conclusions and further research directions.

2. Previous work

Virtual fitting rooms have been a research subject for more than a decade. Protopsaltou et al. [7] developed an Internet-based approach for virtual fitting rooms, although it was not real time

☆This article was recommended for publication by V. Interrante.
* Corresponding author. Tel: +90 312 290 1386; fax: +90 312 266 4047.
E-mail addresses: gultepe@cs.bilkent.edu.tr (U. Gültepe), gudukbay@cs.bilkent.edu.tr (U. Güdükbay).

http://dx.doi.org/10.1016/j.cag.2014.06.001
0097-8493/© 2014 Elsevier Ltd. All rights reserved.

and required marker-based motion capture systems for animation. Zhang et al. [8] used a multi-camera system utilizing shell fitting space (SFS) [9] techniques to build a real time intelligent fitting room.

Advances in time-of-flight technology made depth sensors available at consumer-level prices with better performance. This prompted a wave of research based on depth sensors in various fields, such as rehabilitation [10], indoor modeling [11], and medicine [12]. Another topic that attracted significant attention from both researchers and companies is real-time virtual fitting rooms [13]. Giovanni et al. [14] developed a virtual try-on system utilizing a calibrated set of Kinect and high definition cameras, while comparing the two state-of-the-art depth sensing software development kits (SDKs)-OpenNI [15] and Kinect for Windows SDK [16]. While most frameworks utilize garment meshes with physics simulation [1,2], another intriguing approach is using a pre-recorded apparel image database from which the images are superpositioned onto the RGB video of the user [17,18].

One problem with depth sensors is the feeble quality and noisiness of the depth stream. This problem is analyzed in depth by Khoshelham and Elberink [19], who concluded that the standard deviation reaches 2 cm in a measuring distance of 3 m. Matyunin et al. [20] attempted to improve the quality by filtering with additional information from the attached RGB camera. To increase the quality of the models produced using depth data from a Kinect camera, Tong et al. [21] describe a scanning system for capturing 3D full human body models utilizing multiple Kinects to be used by virtual try-on applications.

A key purpose of both virtual and real fitting rooms is giving the customer the look and feel of clothing of a specific size on the user's body, so the user can choose the appropriate size for him. Embedding the feature of matching clothing sizes with users requires capturing the users' body dimensions. More advanced frameworks even construct virtual avatars with input from only one depth sensor [22,23]. On the other hand, although these works provide higher detail avatars and more precise measurements, which might be more suitable for a made-to-measure type of framework, these processes require too much time to work with a real-time 'fixed-size try-on' virtual fitting room application, and we suggest that simple body height and shoulder width measurements are sufficient. These applications require a faster approach along with a specialized garment design framework such as the works of Yasseen et al. [24] or Meng et al. [13].

There are also notable studies for made-to-measure technologies for online clothing stores [25], shape control techniques for automatic resizing of apparel products [26], modeling a 3D garment on a 3D human model by 2D sketches [27], and garment pattern design using 3D body scan data [28]. Guan et al. [29] describe DRAPE, which is a learned model of clothing that facilitates dressing of 3D virtual humans of different body sizes and shapes with different postures. Their algorithm is composed of three stages: *shape and pose training*, *learning cloth deformation model*, and *virtual fitting*. Brouet et al. [30] present a fully automatic technique to transfer garments between characters with different body shapes in a design-preserving fashion. They formulate garment transfer as a constrained optimization problem and solve it using iterative quadratic minimization. A recent study [31] shows that such applications are well-received by public and have potential commercial uses.

3. The proposed approach

We expect that customized virtual avatars are going to have more significance in the near future in online interactions. To this end, in contrast with the previous work done in this area, our aim is to develop a system that is able to measure the user's dimensions in real time rather than offline and use the measurements to simulate the apparel on a virtual avatar rather than on the RGB image of the user. The results can be used in many applications, ranging from virtual fitting rooms in shopping malls to Massively Multiplayer Online Role-Playing Games with realistic avatars.

The main objective of a virtual dressing room is giving the user the idea of how an item of apparel will look on him/her without actually trying it on. As in any simulation, the absolute reality is almost impossible to attain, although with garment simulation and depth sensing technologies, a substantial level of realism can be achieved. The flow diagram of our framework is given in Fig. 1. The core components of the framework are the following:

- 3D scene rendering and render cycle management,
- weighted skinning and skeletal animation, and
- depth sensor integration.

Because these topics are considered as boiler plate for such an application, they are not explained in detail. We describe the embedded physics engine and explain its functions within the overall framework. We also explain in detail some of the features developed specifically for this particular application.

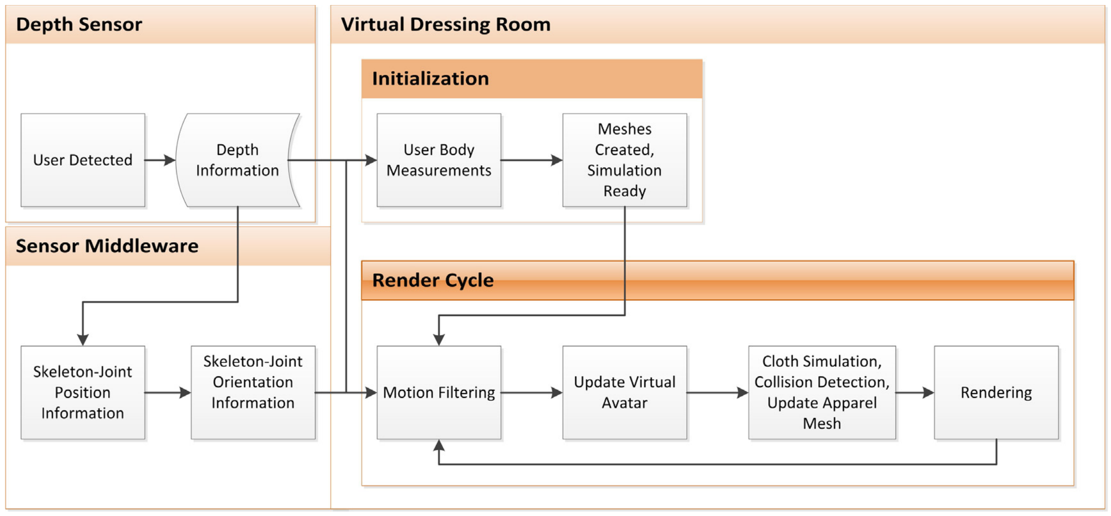

Fig. 1. The overall virtual dressing framework

3.1. Physics engine

The cloth simulation framework utilizes the proprietary PhysX engine by Nvidia [32], with two major components being the *cloth deformation* and *collision detection* modules.

The deformation algorithm is based on the position-based dynamics approach [33], calculating the position of the particles from their previous positions and applying constraints on mutual distance and angle. The approach is stable and efficient for real time applications.

Collision between the body and the cloth is detected using collision spheres and capsules (composed of two spheres) co-located with body joints and bones, respectively (cf. Fig. 2). The body bone information is extracted from the input skeletal mesh and the colliding capsules are generated automatically after the measurement process, with the capsule pair indices stored in an array.

The solution is performed for the trajectory of the capsule or sphere and the particle for each frame interval. This approach is especially robust in fast motion, which is important because the motion is created in real time. The resulting equation is a sixth degree polynomial, which is approximated with a quadratic equation. The solution is performed on the GPU, which increases the parallel performance greatly, allowing for frame rates higher than 600 frames per second (fps). The cloth is discretized as a triangle mesh and the collision is only detected on vertices. The density must be high enough in order to avoid penetration at locations further from the vertices [32,34]. This issue can be resolved using virtual particles for apparel meshes with lower resolution, where additional vertices are introduced at runtime for collision simulation [32].

The acquired body measurements are used to estimate the locations of joints and bones, for collision sphere placement, whereas the radii are used directly for determining the sphere sizes. With both parameters scaled according to the user, various body types can be simulated realistically.

3.2. Body measurements

For a realistic fitting experience, another requirement is acquiring a set of simulation parameters from a human test-subject for a pre-modeled apparel mesh, which is to be displayed on a virtual avatar reflecting the body characteristics of the aforementioned subject. The set of parameters for the simulation includes the body height and the shoulder width, also the radii for the collision spheres, which have their centers coinciding with the joints of the virtual avatar's skeleton. The body width and height are then utilized to estimate the body size of the user, and collision spheres are used in the dressing room simulation, to detect and resolve collision with the cloth particles.

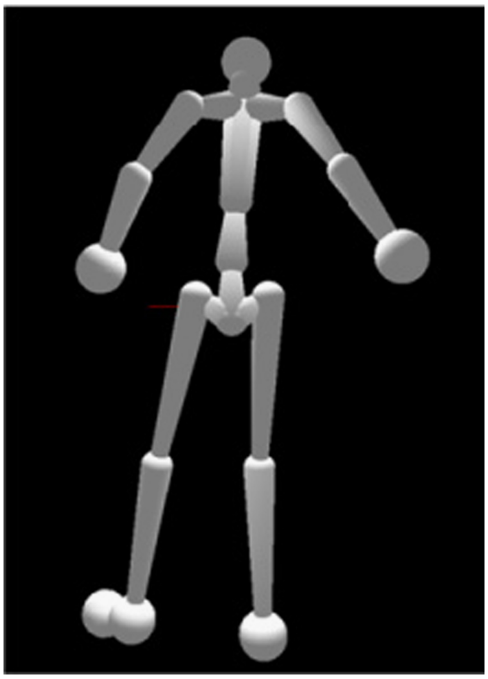
Fig. 2. The colliding human body with the default radii.

As opposed to the studies focusing on acquiring high-detail avatars of subjects for made-to measure applications, we focus on the speed of the overall algorithm for real-time purposes and acquire enough measurements for a ‘fixed-size try-on’ system. By fixed-size try-on, we mean appropriately customizing the avatar in two dimensions using a few body parameters such as the body height and shoulder width. Such a customization works well for standardized body shapes, however, for extreme body shapes, more parameters are needed to experience a realistic fitting experience.

3.2.1. Depth map filtering

The state-of-the-art time-of-flight (TOF) cameras still provide low resolution and quality output compared to the state-of-the-art advanced RGB systems. The quality of the input depth map is a crucial factor in the overall performance of the system. Therefore, we first wish to improve the quality of the depth map by applying canonical image enhancement methods.

We utilize a TOF camera running with a middleware that provides a subject map. The subject map has the same size and origin as the depth map. Each pixel either belongs to the subject or to the background.

Let us take the input depth map D as an $M \times N$ matrix. Initially, the user pixels from D are extracted by a pixel-by-pixel comparison with the input user map. We are only interested in one subject and D_1 represents the depth pixels of the subject, whereas U_1 is the bit map of the subject. Also, the non-subject pixels are set to the mean value of the user pixels to set the matrix properly for the subsequent filterings:

$$D_1 = (D-(D \times U_1)) \times \frac{1}{n} \times \sum_{i=1}^{n} (D \times U_1)_i + D \times U_1, \quad (1)$$

where n is the total number of pixels belonging to the user. The subject depth map is now prepared to be processed with the Gaussian filtering to normalize and improve the quality:

$$D_G = D_1 \ast G, \quad (2)$$

where ‘$\ast$’ is the convolution operator. The Gaussian filtering completes the enhancement of the input depth map. The overall process is described in Algorithm 1.

Algorithm 1. Depth map filtering.

```
   Input: Raw depth and subject stream from depth sensor
   Output: Optimized subject map
1  $depth_{sum} = 0$
2  $n_{user} = 0$
3  for $i$ from 0 to $d_{width}$ do
4      for $j$ from 0 to $d_{height}$ do
5          if $U(i,j)$ then
6              $depth_{sum} = depth_{sum} + D(i,j)$
7              $n_{user} += 1$
8  $depth_{average} = depth_{sum}/n_{user}$
9  for $i$ from 0 to $d_{width}$ do
10     for $j$ from 0 to $d_{height}$ do
11         if not $U(i,j)$ then
12             $D(i,j) = depth_{average}$
```

```
13  for $i$ from 0 to $d_{width}$ do
14      for $j$ from 0 to $d_{height}$ do
15          if $U(i,j)$ then
16              $D(i,j) = D(i-m : i+m, j-n : j+m) * Gaussian(m,n,e)$
17  return $D$
```

3.2.2. Body measurement

By now, we have an optimized depth map, which is ready for performing key body dimension measurements. The key dimensions are handled in two groups: the collision sphere radii, which are used for the collision detection in the simulation, and the height and width parameters, which are used to determine the size of the apparel.

Collision sphere radii: The simulation framework utilizes collision spheres and capsules instead of arbitrary geometries. This constraint allows the simulation to run in real time with exceptionally high frame rates by simplifying the algorithms. There are a total of 15 joint locations provided by the middleware, as shown in Fig. 3. These are the key points for collision sphere placement. For this reason, the framework also utilizes 15 collision spheres and the capsules formed by pairs.

For a realistic simulation, collision spheres must be as large as possible without intersecting the skin mesh of the avatar. The optimal sphere fitting algorithm starts with an infinitely small circle and expands it discretely until it intersects with the body contour. The steps of this algorithm are as follows:

1. Take vector J_i, which represents the coordinates of the ith joint. Initialize the radius of the sphere by setting it to the z-distance with the overlapping point in the depth map:

$$r_i^z = J_i^z - D^z(J_i^x, J_i^y) \tag{3}$$

2. Start with an infinitely small line segment parallel to the x-axis. Expand it until it intersects with the body contour. Take the x-distance between the intersection point and the joint location. Repeat the same process with a line segment parallel to the y-axis and take the larger radius. While expanding the segment, stop expanding and discard the corresponding result if the border of the depth map is reached:

$$r_i^{x,y} = \max(\| \pm J_i^{x,y} \mp D^{x,y}(J_i^{y,x}, J_i^z) \|) \tag{4}$$

3. Take the minimum of the radii of the three axes because there should be no intersection with the body contour and the shape must be a sphere:

$$r_i = \min(r_i^{x,y,z}) \tag{5}$$

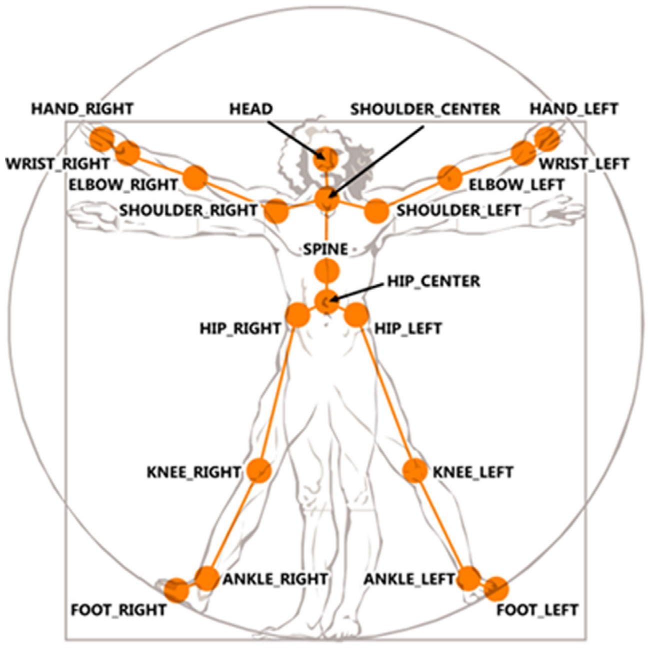

Fig. 3. Human joints provided by Kinect SDK [35].

The pseudocode of the sphere fitting is given in Algorithm 2.

Algorithm 2. Sphere fitting.

```
Input: Optimized depth stream from depth sensor
Output: Collision sphere radii for each joint
1  foreach joint do
2      $p = pos_{J_m}$
3      $r_z = \sqrt{P_z^2 - D_z(P_x, P_y)^2}$
4      for $i$ from $P_x$ to 0 do
5          if $D(i, P_y)$ equals $P_z$ then
6              $r_x^- = i$
7              break
8      for $i$ from $P_x$ to $depth_{width}$ do
9          if $D(i, P_y)$ equals $P_z$ then
10             $r_x^+ = i$
11             break
12     for $j$ from $P_y$ to 0 do
13         if $D(P_x, j)$ equals $P_z$ then
14             $r_y^- = j$
15             break
16     for $j$ from $P_y$ to $depth_{height}$ do
17         if $D(P_x, j)$ equals $P_z$ then
18             $r_y^+ = j$
19             break
20     $r_m = min(r_z, r_x^-, r_x^+, r_y^-, r_y^+)$
21 return $(r_0, r_1 \dots r_n)$
```

Height and width parameters: The width and the height of the subject are important for determining the proper actual size for the cloth. However, a straightforward estimation of the body height and the shoulder width is prone to errors due to the noise and quality of the incoming depth map. In order to minimize this error, an upscaled version of the subject's body is measured, to be used in width-height estimation later. We use the human body proportions defined by [36], which are shown in Table 1, and the primary body parameters defined in Table 2 for different clothing types. For the sake of relative representation, the width and the height of the head are taken as unit width and height, respectively.

Table 1
Human body proportions. The numbers in parentheses correspond to the lines in Fig. 4.

Distance	Width	Height	Measurement source
Head	1w (1)	1h (2)	Depth map + Joint location
Body height	–	7h (3)	Depth map
Hip height	–	4h (4)	Joint location
Elbow to fingertip	–	2h (5)	Depth map + Joint location
Wrist to fingertip	–	1h (6)	Depth map + Joint location
Shoulder width	3w (7)	–	Depth map + Joint location
Hip width	– (8)	–	Depth map
Torso height	–	– (9)	Joint location

Table 2
Primary proportions for different clothing types.

Type of clothing	Primary height proportions	Primary width proportions
Trousers	Hip height	Hip width
Long sleeves	Body height	Elbow-fingertip height, shoulder width
Short sleeves–sleeveless	Torso height	Shoulder width

The measurement source column in the table represents the source for the estimation of the respective parameter:

- The joint location represents that the measurement will take the input subject joint locations as the reference.
- The depth map represents that the measurement will be based on the pixel distribution in the filtered subject depth map.

The depth map and the joint locations are often used together for better performance. Note that some of these parameters are not standard to be used as relative references, such as the hip width. These parameters do not affect others in the estimation process and vice versa (Fig. 4).

The measurements are performed in the real-world space, rather than the projective space, because a real size estimation is crucial for determining the appropriate clothing size. The latter would be sufficient for simulation purposes, when there is no concern of real-life apparel fitting. After the acquisition of the required scaling parameters for the subject, the clothing and avatar should be scaled as a whole in three dimensions uniformly, in contrast to a segmented scaling. This decision is based on the scope of our work because it is a standard-sized apparel fitting application without extensive customization. The clothes are resized based on standard clothing sizes, from XS to XXL. Although there are a variety of standardized clothing sizes across Europe, Asia, USA and the rest of the world, there is not a strict regulation; hence, the exact size ranges could be determined according to the specific company and apparel piece. We use a generic chart provided by a sports goods manufacturer [37]. Following the measurements, the body height and the shoulder width are estimated for cloth resizing as described below (cf. Algorithm 3):

1. Take the primary dimension (either the body height or shoulder width) P_i^0. This process is repeated for width (W) and height (H).
2. Using the remaining measurements, estimate the primary dimension P_i as P_i^j using proportion R_i^j from Table 1:

$$(W,H)_i^j = (W,H)_j \times R_i^j, \quad j = 1, \ldots, N \tag{6}$$

3. Find the optimized primary dimension as the mean of all estimations:

$$(W,H)_i = \frac{1}{(n+1)} \times \sum_{j=0}^{n} (W,H)_i^j \tag{7}$$

Algorithm 3. Body dimension estimation for cloth resizing.

```
    Input: Human body proportions and user and depth data
    Output: Body height and shoulder width
1   $t_{proportion} = \text{import (Table 1)}$
2   $t_{primary} = \text{import (Table 2)}$
3   $ct = cloth_{type}$
4   $width_{main} = t_{proportion}.width(ct)$
5   $width_{sum} = 0$
6   $count_{effector} = 0$
7   foreach $width$ in $t_{proportion}$ do
8       $w_i = measure(p_i)$
9       $w_i^j = w \times t_{proportion}.ratio(p_i, parameter_{main})$
10      $width_{sum} = width_{sum} + w_i^j$
11      $count_{effector}{+}{+}$
12  $width_{weighted} = \frac{width_{sum}}{count_{effector}}$
13  $x_s = \frac{width_{weighted}}{width_{cloth}}$
14  $height_{main} = t_{proportion}.height(ct)$
15  $height_{sum} = 0$
16  $count_{effector} = 0$
17  foreach height in $t_{proportion}$ do
18      $h_i = measure(p_i)$
19      $h_i^j = h \times t_{proportion}.ratio(p_i, parameter_{main})$
20      $height_{sum} = height_{sum} + h_i^j$
21      $count_{effector}{+}{+}$
22  $height_{weighted} = \frac{height_{sum}}{count_{effector}}$
23  $y_s = \frac{height_{weighted}}{height_{cloth}}$
24  return $(x_s, y_s)$
```

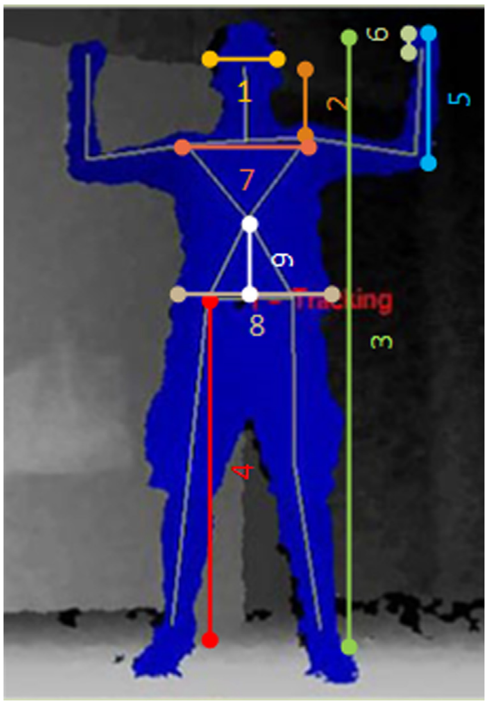

Fig. 4. Human body proportions.

Algorithm 4. Temporal averaging.

```
    Input: Raw depth stream from the depth sensor
    Output: Final collision sphere radii and body dimensions
1   $s = 2 \times 30$ Array for $x$ and $y$ scaling parameters for 30 frames
2   $r = 16 \times 30$ Array for joint radii for 30 frames
3   for $i$ from 0 to 30 frames do
4       $r[i] = fitSpheres()$
5       $s[i] = optimizeScaleParameters()$
6   $r_{final} = avg(r)$
7   $s_{final} = avg(s)$
```

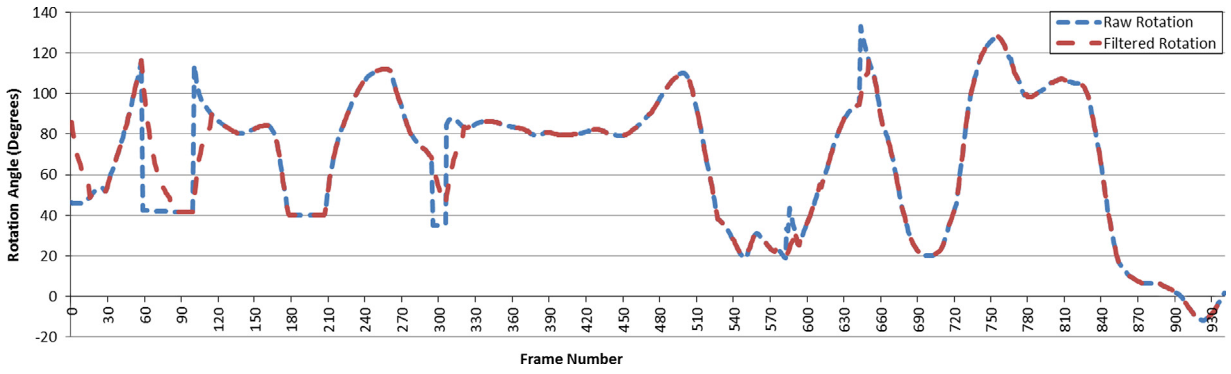

Fig. 5. The raw and filtered samples for the right humerus roll angle.

3.2.3. *Temporal averaging*

By now, we have acquired the required body dimensions and collision sphere parameters for a realistic simulation. Yet, the measurements are performed on a filtered version of a depth sensor with high error rates. In order to overcome the noise and overall depth-sense faults, the prior measurements are repeated for the duration of 1 s, which corresponds to 30 frames of input depth map. A considerably different approach here would be to employ temporal averaging on the depth map instead of temporal averaging on the measured parameters. We observe that the results suffer due to the motion of the subject because most subjects fail to keep their exact form for 1 s. To overcome these problems, we use temporal averaging that takes the mean of the specified parameters for the frames in 1 s (cf. Algorithm 4). This step finalizes the parameters and delivers the required parameters for simulation environment synthesis.

3.3. *Motion smoothing*

Our framework utilizes a virtual 3D avatar to display and simulate the apparel models. The avatar imitates the motions of the actual user, using the orientation data acquired from the depth sensor middleware. However, the application of the raw data from the sensor causes unnatural movements due to the noise in the sensor input, self-occlusions of the body and inadequate Inverse Kinematics (IK) solvers. In order to present a more realistic avatar animation, a series of filters and constraints are applied to the sensor data.

3.3.1. *Position filtering*

The most severe disruption of the self-occlusion problem takes place in the joint position acquisition. There is no possible way of acquiring the correct position of a limb when the sensor has no vision of it. However, the way humans move their limbs under normal conditions follow certain principles and trends, which can be used to estimate the locations of occluded body parts. The nature of these motions, demonstrating traits similar to seasonal behavior, makes them suitable for applying a variety of filters [38]. We decided to go with the Holt–Winters double exponential smoothing [39,40] as it comes with the middleware, is easy to use, and delivers good quality results with acceptable latency for the purposes of this application.

3.3.2. *Rotation filtering and constraints*

The joint orientations are acquired from the sensor middleware directly, however the data is not smooth. Although the middleware enforces certain constraints (such as allowing only pitch rotations on ulna), there are often significant gaps in the estimated angles that produce unnatural tremor-like movements on the avatar. Furthermore, there is no filtering of unnatural rotations that take place when an occluded body part is estimated to be in a wrong location.

The inferior quality of the orientation data is improved in two stages: applying another set of constraints on the joint data based on the natural limits of human bones, followed by an asymptotic smoothing of the joint angles to prevent the effect of gaps in the angles (see Fig. 5).

3.3.3. *Bone splitting*

The lower sections of human limbs contain two parallel bones allowing a twisting rotation of the hands and feet. The configurations of bones allow all portions of the lower limbs to follow the bones in pitch and roll rotations, although the effect of yaw rotation decreases as it gets closer to the mid-section joint (elbow or knee). This effect is not possible to achieve with a single bone fore-arm representation, as specified in the highest level of detail in the H-ANIM standard [41], using unified weights (same for all types of rotations, transformations and scaling) and linear skinning. On the other hand, applying a different set of weights for each possible transformation, rotation or scaling requires additional space and time, which can be considered redundant because it is not going to be used in most parts of the skeleton and surface mesh; thus, it is not implemented in most of the popular rendering engines. This problem is addressed by Kavan et al. [42], proposing a method of introducing additional blending bones to simulate non-linear skinning. However, this approach is not suitable for a real-time application with previously unknown motions.

Another solution to the same problem is presented by Mohr and Gleicher [43], where they automatically insert additional joints in the skeletal system if the existing structure is deemed unfit to achieve the desired skin deformation. They also use the same method to simulate bulges in the skin through muscle contractions. On the other hand, their study aims to solve this problem in the design stage and in an offline manner, rather than at the animation stage of an online system.

Because the problematic bones for this particular case are in the upper limbs, we employed an approach similar to the one described in [43] to solve the problem. We introduce an additional bone connected in series for the upper limbs. The humerus and ulna bones are split halfway and the lower sections are labeled as humerus-extension and ulna-extension. The vertex weights in the corresponding sections are divided linearly among two sections, as seen in Fig. 6.

During runtime, the filtered local rotation of the upper limb bones is separated into two distinct rotations, one containing the yaw and the other containing the pitch and roll rotations, which are applied to the extension bone and the original bone,

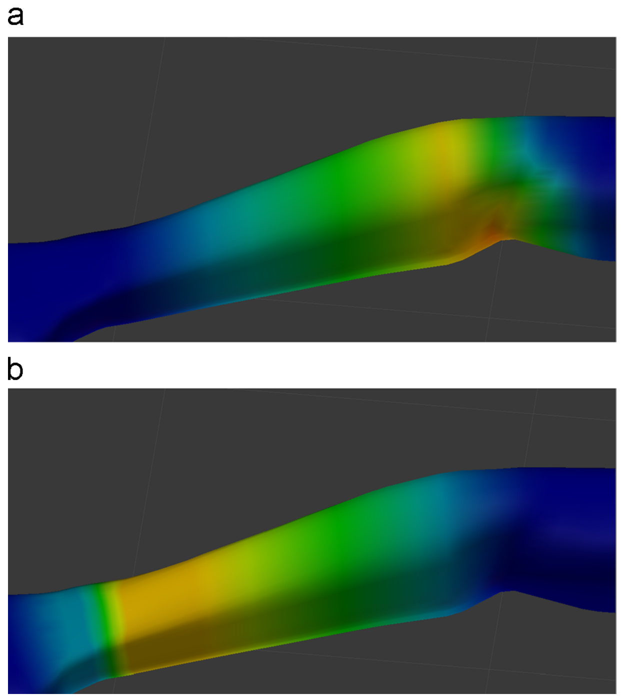

Fig. 6. The vertex weights for (a) the upper ulna and (b) the lower ulna bone (weight increases from blue to yellow). (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

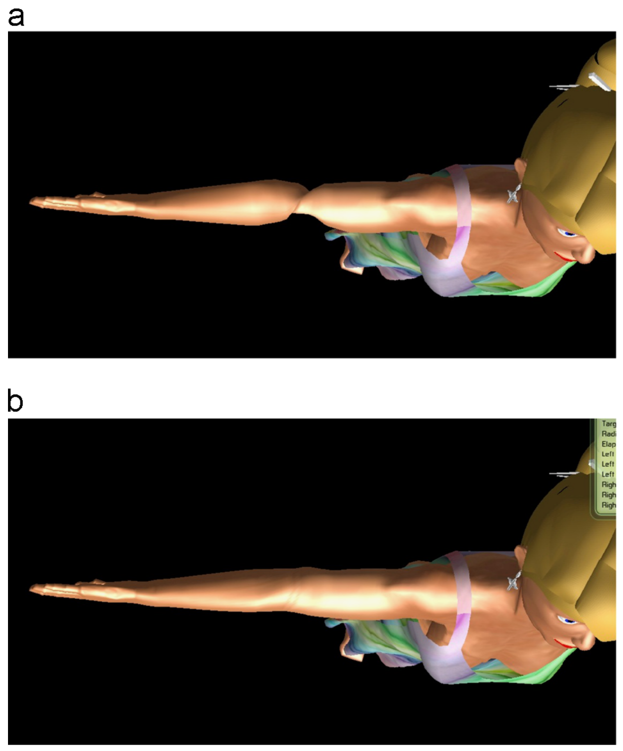

Fig. 7. Comparison of a -90° yaw rotation on the forearm with: (a) single- and (b) double-boned skinning.

respectively. The rotation of the user's arm is transferred to the virtual avatar naturally without any artifacts by the use of proper weights; these weights are derived from the joint-vertex influence weights automatically assigned to a single bone [44] and distributed linearly between the elbow and wrist joints. As seen in Fig. 7, the vertices on the forearm twist in a more natural way resembling the real motion.

3.3.4. *Handling the foot skating problem*

The virtual model is animated by changing bone orientations and root bone position. When the orientations and root position are applied to the bones, the feet of the avatar appears to be floating on the ground, and that deteriorates the realism. This problem, known as footskating, is not limited to depth sensor applications. It shows up in motion-capture-based animation systems and is solved in a reasonable way with approaches such as those proposed by Kovar et al. [45] and Ikemoto et al. [46]. However, these methods rely on some sort of preprocessing or supervised learning, which is not suitable for our system, as the motion is captured and applied in real time. Hence, a solution that does not require a training process is needed for such applications. Sung [47] proposes a similar method that does not require training for real-time depth sensor applications. Their approach is adopted into our framework to overcome the footskating problem, along with additional filters and constraints in order to conform with the nature of our system. It is assumed that one foot is always on the ground; otherwise, artifacts may occur, for example, if the user attempts jumping. The footskating correction method consists of two parts: *constraint checking and adaptation* (cf. Algorithm 5) and *joint angle determination and application* (cf. Algorithm 6). The whole process consists of the following four steps:

Algorithm 5. Constrained foot determination.

```
if $isFirstFrame$ then
    if $p^y_{left} < p^y_{right}$ then
        $constrained = left$
        $y_{threshold} = p^y_{left}$
    else
        $constrained = right$
        $y_{threshold} = p^y_{right}$
if $constrained = left$ then
    // Check if constraints still hold
    if $p^y_{left} > y_{threshold}$ and $v_{left} > v_{threshold}$ then
        // Foot not constrained anymore, check other foot
        if $p^y_{right} > y_{threshold}$ and $v_{right} > v_{threshold}$ then
            // Neither foot are constrained
            $updateThresholds()$
            $restart$
        else
            //Right foot is constrained, feet switched
            $constrained = right$
            $recordRightFootPosition()$
            $constraintSwitched = True$
else
    // Check if constraints still hold
    if $p^y_{right} > y_{threshold}$ and $v_{right} > v_{threshold}$ then
        // Foot not constrained anymore, check other foot
        if $p^y_{left} > y_{threshold}$ and $v_{left} > v_{threshold}$ then
            // Neither foot are constrained
            $updateThresholds()$
            $restart$
        else
            // Left foot is constrained, feet switched
            $constrained = left$
            $recordRightFootPosition()$
            $constraintSwitched = True$
```

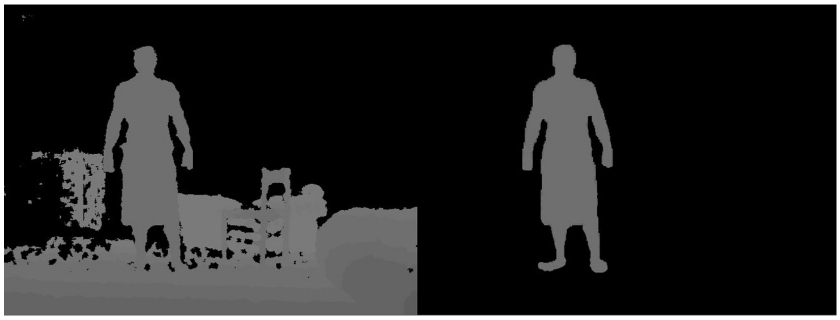

Fig. 8. The effect of depth map filtering. Notice the smoothened user boundary in the right image.

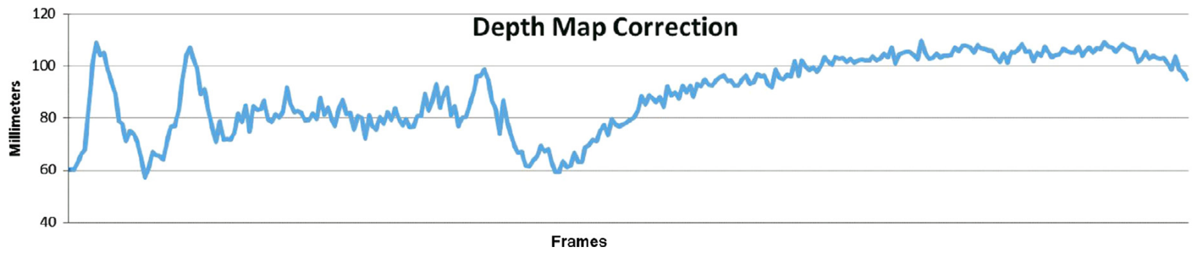

Fig. 9. The average depth map correction per frame over 1 s.

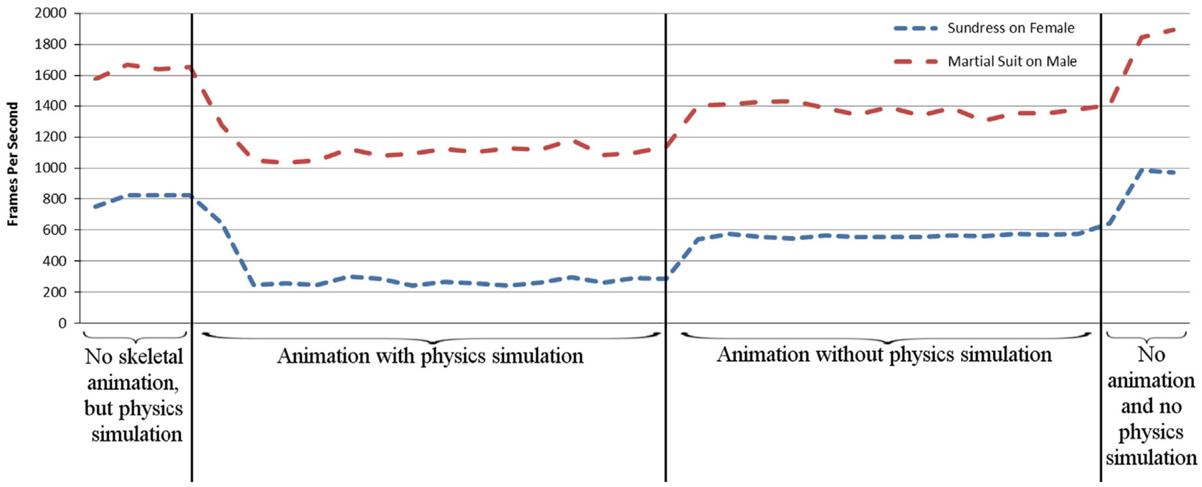

Fig. 10. The frame rates for two simulations with different apparel meshes.

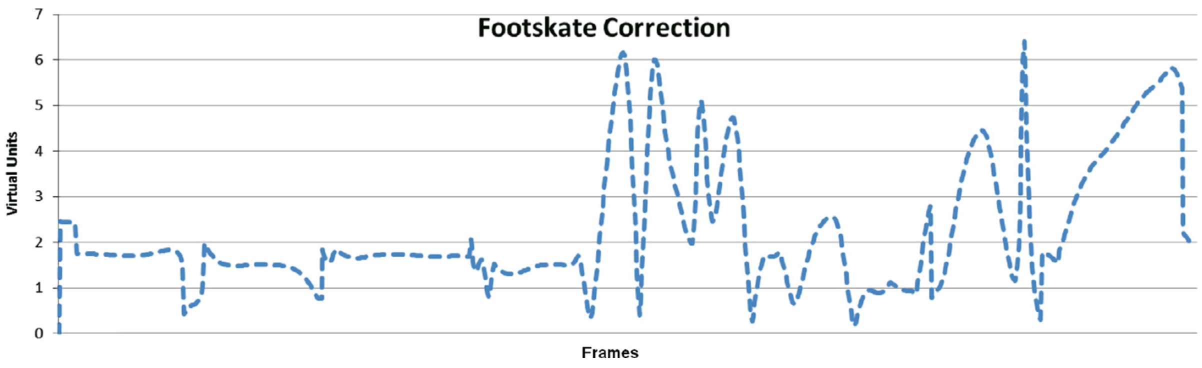

Fig. 11. The corrected displacement of feet. The local minima correspond to changes in the constrained foot and subsequent position smoothing. The zig zag regions correspond to time intervals where the user is performing body yaw motion and the feet are sliding considerably.

Algorithm 6. Foot skating filtering.

```
1 // Check which foot is constrained
2 $checkFootConstraints()$
  $p^{foot}_{constrained} = getRecordedFootPosition()$
3 $target_{ik} = p^{foot}_{constrained} - p^{hip}_{constrained}$
4 // Get rotation quaternions for constrained hip and knee
5 $q_{hip}, q_{knee} = solveIkan(constrained,\ target_{ik})$
6 $hip_{constrained}.rotate(q_{hip})$
7 $foot_{constrained}.rotate(q_{hip})$
8 $p^{foot}_{new} = calculateForwardKinematics()$
9 // Calculate the root displacement vector
10 $d_{root} = p^{foot}_{constrained} - p^{foot}_{new}$
11 $root.translate(\mathrm{d}_{root})$
12 // Proceed with joint smoothing
```

1. Determine the constrained foot by checking speed and location thresholds. The thresholds are changed adaptively in order to compensate for different sensors and user placements.
2. Place the constrained foot at its last recorded position and solve the inverse kinematics problem to determine the orientations of the hip and knee joints. Solving an inverse kinematic problem is no trivial task. There are various numerical and analytical methods for inverse kinematic problems and many implementations focusing on each. We use the IKAN library [48], which uses the approach described in [49].
3. Check if the foot coincides with its intended position after the inverse kinematic orientations are applied to the bones. If there is a positional mismatch, relocate the root joint to complete the alignment.
4. Smooth the joint orientation difference to avoid jumps from frame to frame. This is especially important when the constraint switches, as the source of the orientations is different in these cases and they do not always overlap. Smoothing parameters are optimized using trial and error; they are not viable for adaptive nature. Smoothing also helps to overcome the self-occlusion and the data-noisiness.

Joint smoothing consists of interpolating between frames, especially when there is a significant change in orientation, which would normally cause a non-continuous animation. Old joint orientations are always recorded in the global coordinate system and updated every frame. After we get the new orientation from the new frame, we calculate the quaternion that would rotate the old orientation to the new one:

$$Q_d = Q_{new} \times Q_{old}^{-1} \tag{8}$$

When we compute the delta quaternion, we get its axis-angle representation. Because we are interested in rotating the joint partially, we build up a new quaternion with the same axis and a fraction of the same angle. The fraction varies with whether the constraint was switched recently:

$$Q_d = Quaternion((x, y, z), \alpha) \tag{9}$$

$$Q_d^* = Quaternion((x, y, z), \alpha/k) \tag{10}$$

The fractional rotations are then applied to the joints to get a smoother movement. Other implicit operations include coordinate system transformations, as we are working with three coordinate systems: Global-Render Coordinate System, Local-Render Coordinate System, and IKAN Coordinate System.

4. Experiments

The proposed depth map enhancement algorithm is implemented on the in-house developed virtual dressing software, which acts as the test bench for experimentation. The measurement process to identify the body parameters starts after a subject is recognized, identified and calibrated for tracking. Following the parameter estimation, the virtual avatar and clothes are created and the simulation starts.

Experiments are performed on a high-end PC with Intel i7-2600 and NVIDIA GeForce GTX560Ti. The total time for the measurement of body parameters and resizing the clothing does not exceed 1.005 s. Considering the time for acquiring 30 frames of input from the depth sensor is 1.0 s, it is safe to say that the measurement algorithm does not introduce any delays for a real time application because it needs to run only once at the beginning of the simulation. The actual simulation runs at 600 fps at 1920×1080 resolution. The body sizes are estimated with error rates less than 4% (except the case of a significantly obese woman, where the shoulder width error reached 9%), which is sufficient for the realism of the simulation and determining the appropriate apparel size.

Fig. 8 shows the result of depth map correction where the left part is the pre-filtered image and the right part is the result of the filtering process. The average depth map correction in units of millimeters can be seen in Fig. 9. The average correction over the

Table 4
Performance comparison with [14] and [50] regarding height measurements.

Method	Error average (%)	Error deviation (%)	Duration	Estimation
Our approach	2.67	2.06	1.00	Fixed
Giovanni et al. [14]	4.00	3.00	1.00	None
Samejima et al. [50]	6.72	4.68	n/a	PCA

Table 3
Performance figures for seven different subjects.

Subject	Shoulder width (cm)				Body height (cm)			
	Real	Estimated	Error (%)	Deviation	Real	Estimated	Error (%)	Deviation
1	43	43.6	1.0	7.8	172	170.8	0.6	2.5
2	46	44.3	3.0	8.5	176	172.1	2.0	1.4
3	51	48.8	4.0	12.1	176	174.0	1.0	6.6
4	48	50.0	4.0	2.5	188	191.0	1.5	7.1
5	44	42.6	3.0	4.2	178	175.8	1.2	3.1
6	48	52.5	9.0	9.5	162	158.4	2.2	6.1
7	52	50.3	3.4	7.4	203	206.2	1.5	6.7

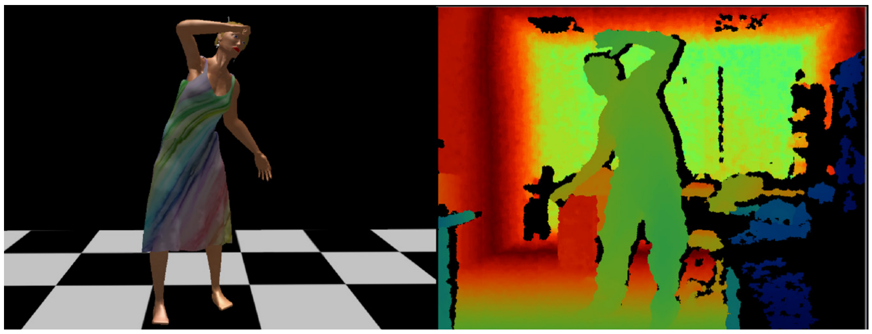
Fig. 12. An example depth map data and the corresponding posture of the subject with a virtual cloth on it (see Video S1 for the virtual fitting room application at work).

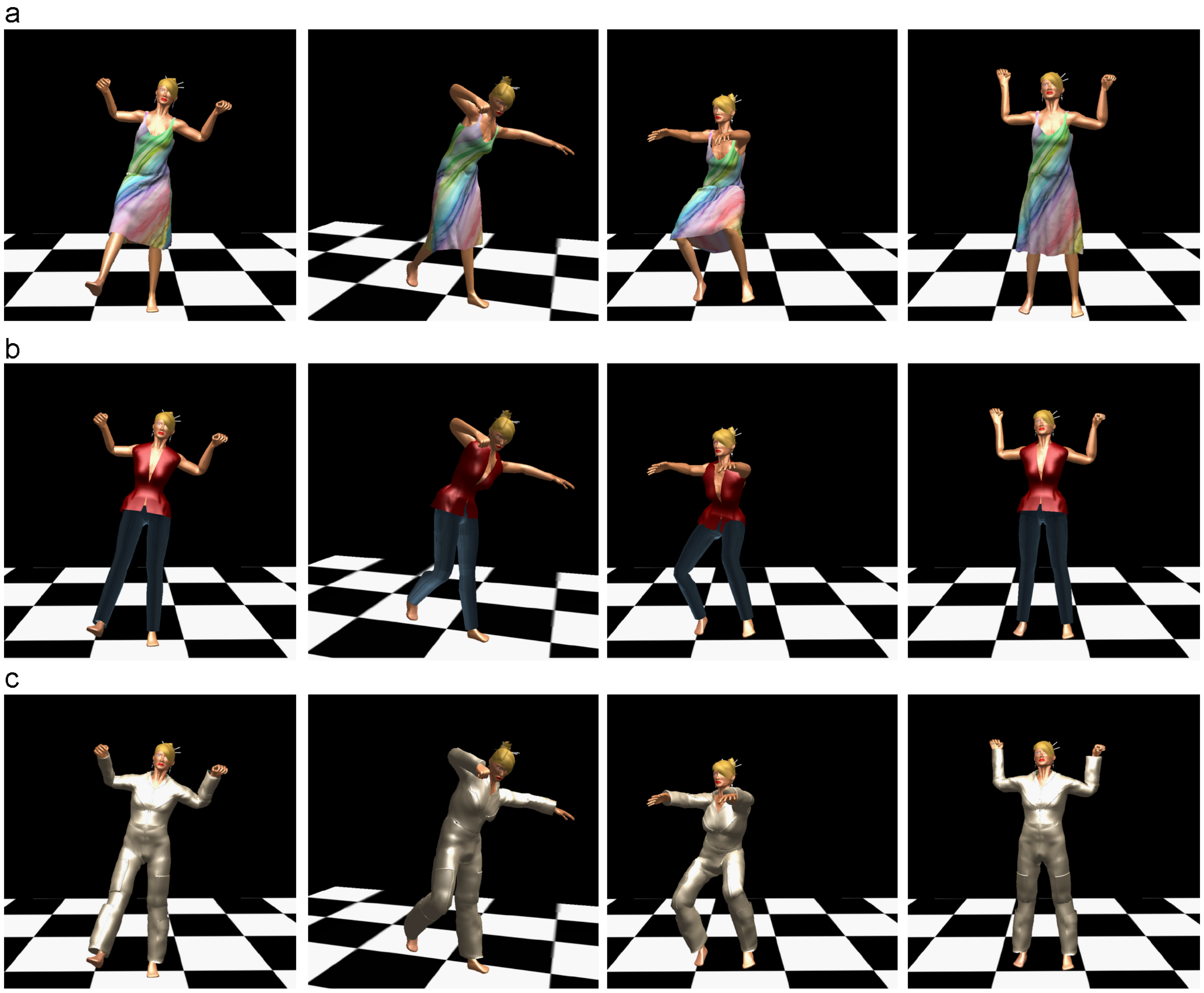

Fig. 13. Examples of different garments on a model with different postures: (a) sun dress, (b) jeans and vest, and (c) flight suit.

time interval is found to be 9 cm. Note that these correction values are collected over the user region, rather than the whole depth image. Nine centimeters, although high, is still expected with the patched holes and Gaussian smoothing over the edges causing erosion. The variation is due to the user not being stable over time, causing different depth distributions.

The frame rates for two simulations with different apparel meshes can be seen in Fig. 10. Notice the common pattern in both of the simulations. The first drop corresponds to the start of the animation of the avatar with the input from the depth sensor. Until that point, the avatar is static, the motion filters do not run and the simulation runs at a higher speed. The frame rate drops approximately 600 fps due to motion filtering and skinning. The first cliff in the graph corresponds to the point where the physics simulation is stopped. The second cliff starts where the input from the depth sensor is stopped and the simulation is stagnant. On average, the physics simulation takes 10.19 ms, so the average physics simulation rate is less than 98 simulations per second.

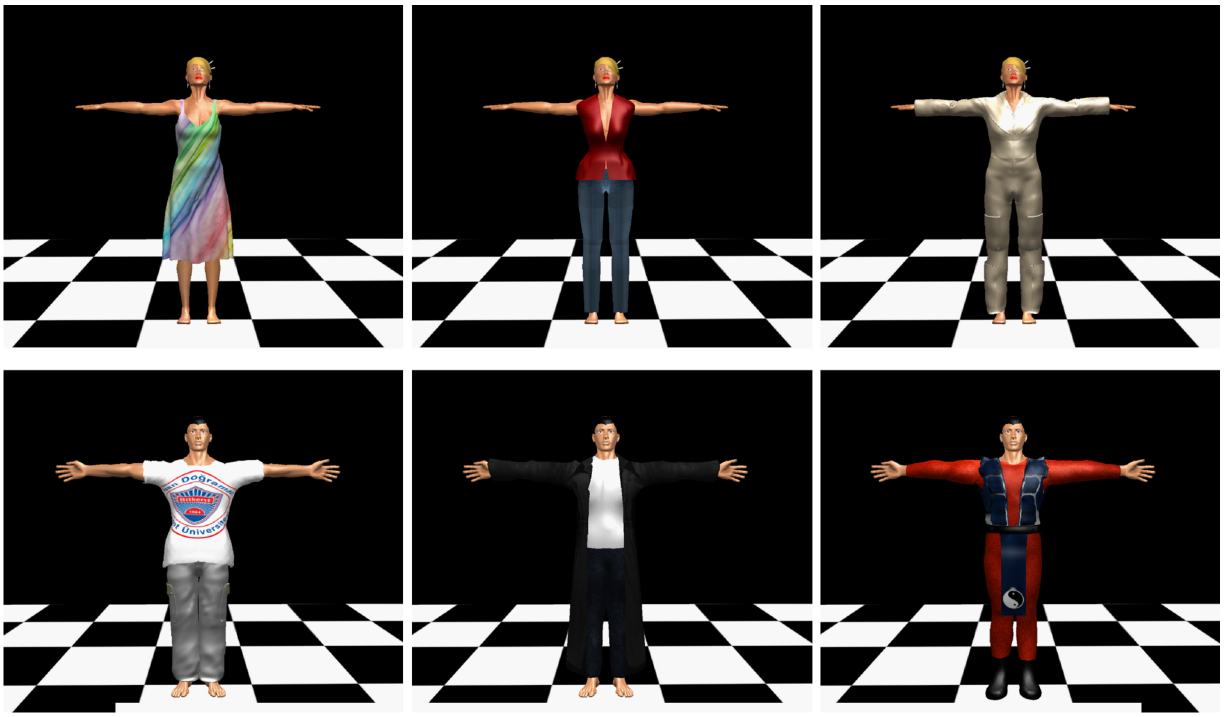

Fig. 14. The designed apparel meshes for the male and female avatars.

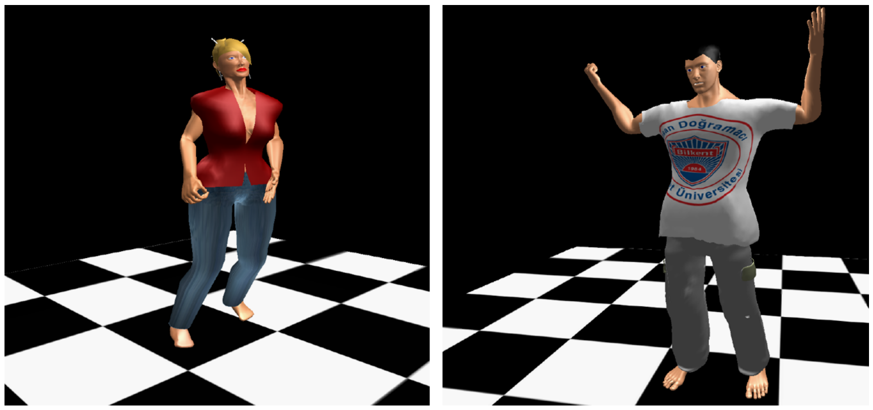

Fig. 15. The results on an obese woman and a tall man.

Table 5
Vertex and face counts of the used meshes. Skinned vertices and triangles are the ones animated with the linear blended skinning, simulated ones are handled by the physics engine.

Mesh	Vertex count		Triangle count	
	Skinned	Simulated	Skinned	Simulated
Female	120,382	–	234,062	–
Male	44,262	–	88,234	–
Duster	3459	480	6660	840
Sundress	1048	3096	1951	6054
Flight suit	21,777	–	42,938	–
Pants–shirt	25,088	–	49,464	–
Jeans–vest	6024	–	11,930	–
Martial suit	5512	778	10,968	1384

After a physics simulation is complete, no new simulation is started for at least 8 ms, to keep the frame rate high for motion filters, which take on average 0.52 s and run every frame.

We compare the simulation performance with that of Meng et al. [13] in terms of cloth mesh complexity and frame rate with similar timestep values. They obtained a frame rate of 63.07 fps for a wawa-style blouse clothing mesh containing 2398 vertices and 4768 triangles. We obtained a frame rate of 268.3 fps for a sundress clothing mesh containing 3096 vertices and 6054 triangles. We should state that our machine is significantly more powerful than theirs.

The filters on the legs focus on solving the foot skating problem. The corrected displacement values can be seen in Fig. 11. The virtual unit is the unit step in the virtual world. For reference, the virtual avatars' height is 55 virtual units. Hence, a virtual unit approximately corresponds to 3.5 cm.

Table 3 presents the measurements of the body dimensions, as well as the errors and standard deviations of the measurements in 30 frames for seven subjects. Our results are compared with the results from [14] and [50], which are also real-time applications, in Table 4.

For the collision spheres, the quality of the results can be assessed by the smoothness of the collision simulation, as seen in

Fig. 12. Throughout the simulation, unnatural intersections between the clothes and the avatar never take place, while the cloth appears to rest on the skin naturally, without space between the two meshes. Fig. 13 shows examples of three garments on a model with different postures generated with our implementation. The six apparel meshes, three for the female avatar and three for the male avatar, can be seen in Fig. 14. Visual results for people with extreme body shapes can be seen in Fig. 15; we provide visual results for an obese woman (Subject 6) and a tall man (Subject 7). Rendering details for the apparel and humanoid meshes can be seen in Table 5.

5. Conclusions and future work

We propose a novel virtual fitting room using depth sensor data. The framework yields a realistic fitting experience for standard body types with customized motion filters, body measurement and physical simulation. The proposed scaling method adjusts the avatar's body size parameters and determines a suitable apparel size, and prepares the collision mesh and the physics simulation, within a total preprocessing time of 1 s. We apply real-time motion filters that prevent unnatural artifacts by smoothing the depth sensor data and estimate the locations of self-occluded body parts. Bone splitting is used to approximate the behavior of double bones in lower arms and legs for realistic animation and rendering of the body parts near the joints.

The most important limitation of the framework is its insufficiency in customization. In its current form, it provides a realistic fitting experience for standard body types. For example, for the obese woman that we experimented, although she has a greater overall body width, she still has a very well-defined waist, which is not realistic for obese people in general. This can be overcome by using more parameters such as waist width. We also expect to be able to get better input quality with advancements in depth sensing technology, hence providing a more realistic experience by increasing the precision. Another limitation is the small simulation zone, which limits the user from seeing the cloth while walking and during other motions. This limitation can be overcome by introducing multiple calibrated sensors.

In future work, we would like to improve the quality of the measurements and visual scaling by using data from an RGB sensor as well, because it provides additional data. We would like to increase the number of collision spheres for better collision detection.

Appendix A. Supplementary data

Supplementary data associated with this paper can be found in the online version at http://dx.doi.org/10.1016/j.cag.2014.06.001.

References

[1] Fitnect Interactive Kft. 3D Virtual fitting dressing room/mirror. Available at ⟨http://www.fitnect.hu/⟩ [accessed on February 2014].
[2] Styku, Inc. Virtual fitting room and body scanning. Available at ⟨http://www.styku.com/⟩ [accessed on February 2014].
[3] FaceCake Marketing Technologies, Inc. Visual demonstration system. Available at ⟨http://www.facecake.com/⟩ [accessed on February 2014].
[4] Lee Yongjoon, Ma Jaehwan, Choi Sunghee. Automatic pose-independent 3D garment fitting. Comput Graph 2013;37(7):911–22.
[5] Li Jituo, Ye Juntao, Wang Yangsheng, Bai Li, Lu Guodong. Fitting 3D garment models onto individual human models. Comput Graph 2010;34(6):742–55.
[6] Fuhrmann Arnulph, Groß Clemens, Luckas Volker, Weber Andreas. Interaction-free dressing of virtual humans. Comput Graph 2003;27(1):71–82.
[7] Protopsaltou D, Luible C, Arevalo-Poizat M, Magnenat-Thalmann N. A body and garment creation method for an internet-based virtual fitting room. In: Proceedings of computer graphics international (CGI '02). Berlin: Springer; 2002. p. 105–22.
[8] Zhang W, Matsumoto T, Liu J, Chu M, Begole B. An intelligent fitting room using multi-camera perception. In: Proceedings of the 13th international conference on intelligent user interfaces (IUI '08). New York, NY, USA: ACM; 2008. p. 60–9.
[9] Cheung K-M, Baker S, Kanade T. Shape-from-Silhouette across time, Part II: Applications to human modeling and markerless motion tracking. Int J Comput Vis 2005;63(2):225–45.
[10] Changa Y-J, Chenb S-F, Huang J-D. A Kinect-based system for physical rehabilitation a pilot study for young adults with motor disabilities. Res Dev Disabil 2011;32(6):2566–70.
[11] Henry P, Krainin M, Herbst E, Ren XF, Fox D. RGB-D mapping: using kinect-style depth cameras for dense 3D modeling of indoor environments. Int J Robot Res 2012;31:647–63.
[12] Gallo L, Placitelli AP, Ciampi M. Controller-free exploration of medical image data: experiencing the kinect. In: Proceedings of the 24th international symposium on computer-based medical systems (CBMS); 2011. p. 1–6.
[13] Meng Y, Mok PY, Jin X. Interactive virtual try-on clothing design systems. Comput Aid Des 2010;42(4):310–21.
[14] Giovanni S, Choi YC, Huang J, Tat KE, Yin K. Virtual try-on using kinect and HD camera. In: Proceedings of the fifth international conference on motion in games (MIG'12), Lecture notes in computer science, vol. 7660. Berlin: Springer; 2012. p. 55–65.
[15] OpenNI Consortium. OpenNI—the standard framework for 3D sensing. Available at ⟨http://www.openni.org/⟩ [accessed on February 2014].
[16] Microsoft, Inc. Kinect for windows. Available at ⟨http://www.microsoft.com/en-us/kinectforwindows/⟩ [accessed on February 2014].
[17] Hauswiesner S, Straka M, Reitmayr G. Virtual try-on through image-based rendering. IEEE Trans Vis Comput Graph 2013;19(9):1552–65.
[18] Zhou X, Shu B, Zhuo S, Deng X, Tan P, Lin S. Image-based clothes animation for virtual fitting. In: SIGGRAPH Asia technical briefs, Article 33. New York, NY, USA: ACM; 2012. 4 pp.
[19] Khoshelham K, Elberink SO. Accuracy and resolution of kinect depth data for indoor mapping applications. Sensors 2012;12:1437–54.
[20] Matyunin S, Vatolin D, Berdnikov Y, Smirnov M. Temporal filtering for depth maps generated by kinect depth camera. In: Proceedings of the 3DTV conference: the true vision—capture, transmission and display of 3D video (3DTV-CON); 2011. p. 1–4.
[21] Tong J, Zhou J, Liu L, Pan Z, Yan H. Scanning 3D full human bodies using Kinects. IEEE Trans Vis Comput Graph 2012;18(4):643–50.
[22] Cui YY, Chang W, Nöll T, Stricker D. KinectAvatar: fully automatic body capture using a single kinect. In: Proceedings of the 11th Asian conference on computer vision (ACCV 2012) workshops, Lecture notes in computer science, vol. 7729. Berlin: Springer; 2013. p. 133–47.
[23] Cui Y, Schuon S, Chan D, Thrun S, Theobalt C. 3D shape scanning with a time-of-flight camera. In: Proceedings of IEEE conference on computer vision and pattern recognition (CVPR); 2010. p. 1173–80.
[24] Yasseen Z, Nasri A, Boukaram W, Volino P, Magnenat-Thalmann N. Sketch-based garment design with quad meshes. Comput Aid Des 2013;45(2):562–7.
[25] Cordier F, Seo H, Magnenat-Thalmann N. Made-to-measure technologies for an online clothing store. IEEE Comput Graph Appl 2003;23(1):38–48.
[26] Meng Y, Wang CCL, Jin X. Flexible shape control for automatic resizing of apparel products. Comput Aid Des 2012;44(1):68–76.
[27] Wang CCL, Wang Y, Yuen MMF. Feature based 3D garment design through 2D sketches. Comput Aid Des 2003;35(7):659–72.
[28] Kim SM, Kang TJ. Garment pattern generation from body scan data. Comput Aid Des 2003;35(7):611–8.
[29] Guan P, Reiss L, Hirshberg DA, Weiss A, Black MJ. DRAPE: dressing any person. ACM Trans Graph 2012;31(4):10 Article 35.
[30] Brouet R, Sheffer A, Boissieux L, Cani M-P. Design preserving garment transfer. ACM Trans Graph 2012;31(4):11 Article 36.
[31] Kim D-E, LaBat K. Consumer experience in using 3D virtual garment simulation technology. J Text Inst 2013;104(8):819–29.
[32] Kim T-Y. Character clothing in PhysX 3. In: ACM SIGGRAPH Asia Nvidia tech talks, Hong Kong; 2011.
[33] Müller M, Heidelberger B, Hennix M, Ratcliff J. Position based dynamics. J Vis Commun Image Represent 2007;18(2):109–18.
[34] Tonge R. Collision detection in PhysX. Recent advances in real-time collision and proximity computations for games and simulations. In: ACM SIGGRAPH course notes; 2010.
[35] Raiten S. Kinect—getting started—become the incredible hulk—code project. Available at ⟨http://www.codeproject.com/Articles/213034/Kinect-Getting-Started-Become-The-Incredible-Hulk⟩; 2011 [accessed on February 2014].
[36] Willis B. Body proportions in art. In: Wunderland web design, 2013.
[37] SKINS International Trading AG. Unique sizing. Available at ⟨http://www.skins.net/en-AU/why-skins/unique-sizing-guide.aspx⟩ [accessed on February 2014].
[38] Azimi M. Skeletal joint smoothing, White paper, Microsoft Corp. Available at ⟨http://msdn.microsoft.com/en-us/library/jj131429.aspx⟩ [accessed on February 2014].
[39] Holt CC. Forecasting trends and seasonals by exponentially weighted averages. Carnegie Institute of Technology, Pittsburgh ONR Memorandum, No. 52; 1957.
[40] Kalekar PS. Time series forecasting using Holt–Winters exponential smoothing. Kanwal Rekhi School of Information Technology; 2004.
[41] H-Anim. Humanoid animation working group. Available at ⟨http://www.h-anim.org⟩ [accessed on February 2014].

[42] Kavan L, Collins S, O'Sullivan C. Automatic linearization of nonlinear skinning. In: Proceedings of the symposium on interactive 3D graphics and games (I3D '09), New York, NY, USA: ACM; 2009. p. 49–56.
[43] Mohr A, Gleicher M. Building efficient, accurate character skins from examples. ACM Trans Graph 2003;22(3):562–8.
[44] Softonic International S.L. Blender 3D Content Creation Suite. Available at ⟨http://blender.en.softonic.com/⟩ [accessed on February 2014].
[45] Kovar L, Schreiner J, Gleicher M. Footskate cleanup for motion capture editing. In: Proceedings of the ACM SIGGRAPH/eurographics symposium on computer animation (SCA'02). New York, USA: ACM; 2002. p. 97–104.
[46] Ikemoto L, Arikan O, Forsyth D. Knowing when to put your foot down. In: Proceedings of the symposium on interactive 3D graphics and games (I3D'06). New York, USA: ACM; 2006. p. 49–53.
[47] Sung M. Automatic fixing of foot skating of human motions from depth sensor. In: Multimedia and ubiquitous engineering. Lecture notes in electrical engineering, vol. 240. The Netherlands: Springer; 2013. p. 405–12.
[48] Human Modeling and Simulation Laboratory, University of Pennsylvania. Inverse kinematics using analytical methods—IKAN. Available at ⟨http://cg.cis.upenn.edu/hms/software/ikan/ikan.html⟩ [accessed on February 2014].
[49] Tolani D, Goswami A, Badler NI. Real-time inverse kinematics techniques for anthropomorphic limbs. Graph Models 2000;62(5):353–88.
[50] Samejima I, Maki K, Kagami S, Kouchi M, Mizoguchi H. A body dimensions estimation method of subject from a few measurement items using KINECT. In: Proceedings of the IEEE international conference on systems, man, and cybernetics (SMC); 2012.