Textured and animatable human body mesh reconstruction from a single image.

Creator` *Levon Khachatryan*

Supervisor` *Shant Navasardyan*

Table of Content

1. INTRODUCTION

We provide a method that will surely excite every Potterhead on the planet. As we all remember, the people in the photos hanging on the fictional wall of Hogwarts freely move around and even jump into other frames to visit. Our method helps to make the magic a reality, so we can make a person in a 2-D photo perform various motions.

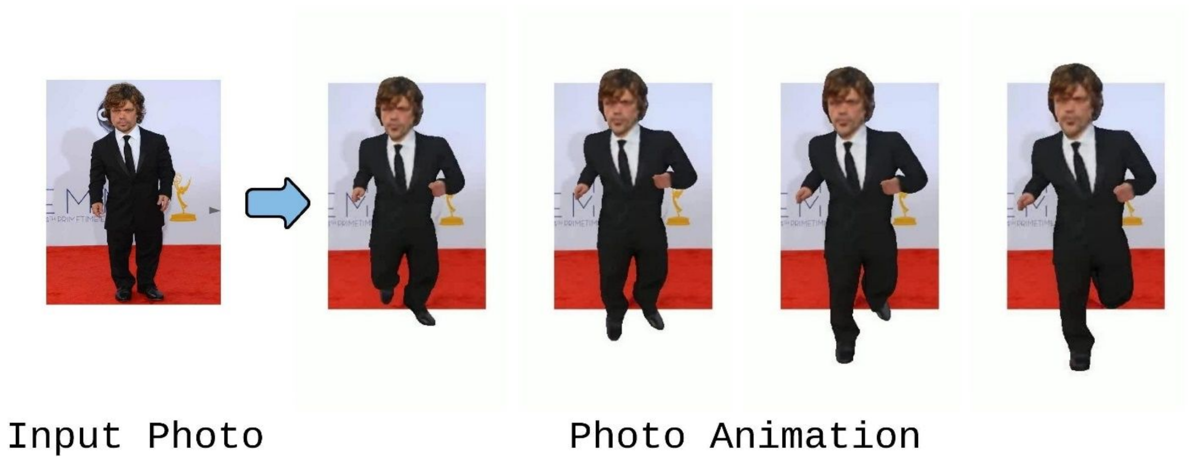

Figure 1: Given a single photo as input (far left), we create a 3D animated version of the subject, which can now walk towards the viewer (middle).

In this project, we address the problem of reconstructing a fully textured and animatable human body mesh from only a single image. It has many applications ranging from virtual and augmented reality to the production of movies and video games. Our contributions are the following:

- we obtain a fully textured and animatable mesh by combining textured mesh and SMPL predictions,
- to the best of our knowledge, this is the first work which makes the predicted fully textured human body mesh animatable.

This work consists of Introduction and five paragraphs. In the first paragraph we introduce some related works. In the second paragraph we describe a method to obtain textured and animatable human body mesh from a single image. In the third paragraph we show some results. In paragraph four we give ideas for future work. Finally, in the fifth paragraph we make a conclusion.

2. RELATED WORK

This paragraph surveys previous work in 3D mesh reconstruction, animation and related techniques. In general there are two approaches for 3D reconstruction: parametric and non-parametric.

A well known parametric method is the skinned multi-person linear model[3], which parameterizes the human body by shape (how individuals vary in height, weight and body proportions) and pose parameters.

As a non-parametric method, the 3D reconstruction in function space[6] was introduced in 2018. This method proposes to consider the continuous decision boundary of a deep neural network classifier as a 3D surface which allows to extract 3D meshes at any resolution.

3D Mesh and Skeleton

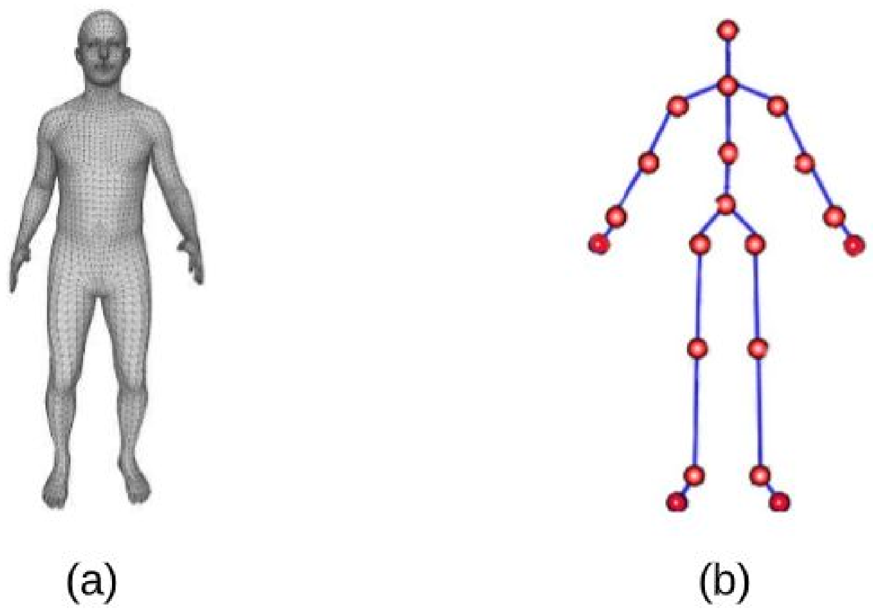

Figure 2: Example of 3D mesh (a), and skeleton (b).

3D mesh (figure 2 (a)), also known as skin, is a collection of vertices, edges and faces that make up a 3D object. Each vertex in the mesh stores x, y, and z coordinate information and can have associated color. Furthermore, two vertices can be connected by no more than one edge. Each face is a triangle consisting of vertices and edges of the mesh. We denote the number of mesh vertices by N.

Skeleton (figure 2 (b)), also known as underlying structure, is a tree, whose nodes are called joints. Joints are the points of articulation we create to control the model. Let K denote the number of skeleton joints.

We will say that a 3D mesh is animatable, if we can give a motion and get the mesh in the corresponding pose. To that end, we need to bind the 3D mesh to the skeleton, so that it moves when we manipulate the skeleton. This process is called a blend skinning, which will be considered in the next, linear blend skinning subparagraph.

Rigging a 3D mesh

Creating an animatable 3D mesh is hard. Once a mesh has been created, and before it can be animated, it must be rigged. This refers to the process of creating a skeleton for the mesh. As we mentioned in the previous subparagraph, skeleton is a tree, so it has a hierarchical structure where each joint is in a parent/child relationship with the joints it connects to. This simplifies the animation process as a whole.

Now, let us define the world coordinate system. It is the right handed cartesian coordinate system where we define the 3D mesh to be displayed. The joints comprising the skeleton are defined in their own local coordinate systems and are assembled into the world coordinate system by a nested series of transformations. These nested transformations ensure that joints inherit the rotations applied to joints higher in the skeleton. A rotation applied at, say, the shoulder joint, causes the entire arm joints to rotate, and not just the upper arm joint. One joint in the skeleton needs to be specified as the root, transformations applied to this joint move the entire skeleton in the world coordinate system. The global transformations applied to any particular joint within the skeleton can be computed by traversing the skeleton from the root to the joint and concatenating the local transformations at each joint visited by the traversal.

There are many methods for reconstructing a 3D mesh, but only a few are able to reconstruct the skeleton along with the mesh. Therefore, the need arises to create a method that reconstructs the skeleton for a 3D mesh. In paragraph three, we present our method for rigging a 3D mesh and application of online service MIXAMO[1]. The latter, applies machine learning to understand where the limbs of a 3D mesh are and to insert a skeleton into the 3D mesh. This is not a fully automatic method, as the user must first place markers on key points (wrists, elbows, knees and groin).

Linear Blend Skinning

Blend skinning is the process of binding a 3D mesh to the skeleton. This means that for each vertex in the mesh, we need to find its associated joints in the skeleton. For example, the vertices making up the polygons in the mesh's thigh would be associated with the joint for the thigh. Vertices of the mesh can normally be associated with multiple joints, each one having scaling factors called **blend weights**. In linear blend skinning[4], the associations between vertices and joints are defined linearly.

Before we formally define the linear blend skinning, let's see what a T-pose and T-pose mesh are. T-pose, also known as rest pose, is a default pose for a 3D model's skeleton before it is animated, and respectively, the T-pose mesh (figure 3) is a 3D mesh in a T-pose.

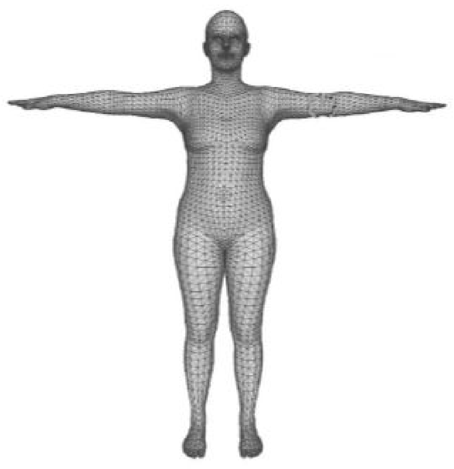

Figure 3: Example of T-pose mesh.

Linear blend skinning assumes the following inputs data:

- **T-pose mesh**, whose vertices are defined as $v_1, v_2, \dots, v_N \in R^3$. It is often convenient to assume that $v_i \in R^4$, with the last coordinate equal to one, according to the common convention of homogeneous coordinates. Please note that these vertices are given in the local coordinate system. During the transformation, the coordinates of the vertex are fixed in the local coordinate system; instead, the position of the local coordinate system relative to the world coordinate system changes.
- **Joint transformations**, represented using a list of matrices $T_1, T_2, \dots, T_K \in R^{4\times4}$. The matrix T_i corresponds to spatial transformation aligning the rest pose of joint i with its current (animated) pose.
- **Blend weights.** We denote the blend weights of v_i vertex as $w_{i,1}, w_{i,2}, \dots, w_{i,K} \in R$.

LBS computes deformation of v_i according to the following formula:

$$v_i' = \sum_{j=1}^{K} w_{i,j} T_j v_i$$

Skinned Multi-Person Linear Model [3]

This is a parametric learned model which reconstructs person 3D mesh from pose and shape parameters:

- the pose parameter $\theta = [\theta_0, \theta_1, \dots, \theta_K]^T$ ($\theta \in R^{72}$) is defined by $3 * 23 + 3 = 72$ parameters; i.e. 3 for each joint plus 3 for the root orientation, (θ_k denotes the axis-angle representation of the relative rotation of joint k with respect to its parent in the skeleton),
- the shape parameter β ($\beta \in R^{10}$) is the first 10 coefficients of a PCA shape space.

For the sake of brevity, a skinned multi-person linear model is often called just SMPL.

For SMPL mesh, faces are fixed, so to reconstruct the mesh we need to find a collection of vertices. This is exactly what the SMPL model does:

$$M(\beta, \theta;\ \Phi) : R^{|\theta| \times |\beta|} \rightarrow R^{3N}$$

In addition to these input parameters, some parameters are learned from data:

- T-pose mesh represented by a vector of N concatenated vertices, $V \in R^{3N}$ (*figure 4(a)*),
- set of blend weights, $W \in R^{N \times K}$,
- blend shape function, $B_S(\beta) : R^{|\beta|} \rightarrow R^{3N}$, (*figure 4(b)*),
- function to predict K joint locations (white dots in *figure 4*), $J(\beta) : R^{|\beta|} \rightarrow R^{3K}$,
- and pose-dependent blend shape function, $B_P(\theta) : R^{|\theta|} \rightarrow R^{3N}$ (*figure 4(c)*).

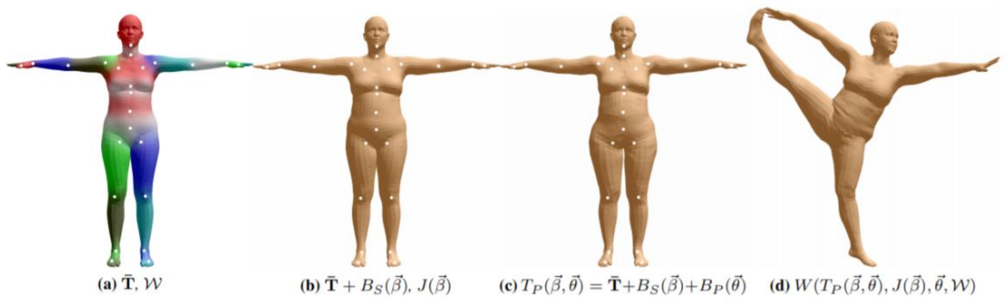
(a) $\bar{\mathbf{T}}, \mathcal{W}$
(b) $\bar{\mathbf{T}} + B_S(\vec{\beta}), J(\vec{\beta})$
(c) $T_P(\vec{\beta}, \vec{\theta}) = \bar{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta})$
(d) $W(T_P(\vec{\beta}, \vec{\theta}), J(\vec{\beta}), \vec{\theta}, \mathcal{W})$

Figure 4: SMPL model. (a) Template mesh with blend weights. (b) shape blend-shape contribution. (c) pose blend-shapes contribution; note the expansion of the hips. (d) Deformed vertices.

And the deformation of v_i will be as follows:

$$v_i' = \sum_{k=1}^{K} w_{i,k} T_k(\theta,\ J(\beta))(v_i + b_{S,i}(\beta) + b_{P,i}(\theta))$$

where $b_{S,i}(\beta)$ and $b_{P,i}(\theta)$ are vertices in $B_S(\beta)$ and $B_P(\theta)$ respectively and represent the shape and pose blend shape offsets for the vertex v_i. These offsets help linear blend skinning work more precisely and take into account the person's shape and pose dependent deformations.

In SMPL, transformation matrices are calculated using not only pose parameter θ, but also joint locations J. To find precise joint locations, SMPL adds shape dependent offsets to default (each default joint is represented by its 3D location in the rest pose).

3D Reconstruction in Function Space [6]

This is a method for 3D mesh reconstruction based on directly learning the continuous 3D occupancy function o. Further this function is approximated with a neural network f_θ.

We will call the function $o : R^3 \rightarrow \{0,\ 1\}$ as an occupancy function of the 3D object (with this function, we can assign to every point $p \in R^3$ an occupancy probability between 0 and 1). Note that decision boundary of function o implicitly represents the object’s surface. If we want our function to predict occupancy probability based on the object's observation (e.g., image) we must condition it on the input. Therefore our f_θ neural network will be as follows:

$$f_\theta : R^3 \times X \rightarrow [0,\ 1]$$

In the process of 3D mesh reconstruction, the marching cubes algorithm[9] is used. Therefore, before delving into the process, let's see what the marching cubes algorithm is.

Marching cubes is a simple algorithm for reconstructing a mesh from a 3D discrete scalar field (also known as occupancy field), whose elements are called voxels. Each voxel has associated status, a value from $\{0,\ 1\}$, defining whether it is inside or outside of the surface whose mesh we want to reconstruct. Algorithm works by iterating ("marching") over a uniform grid of voxels and taking eight neighbor locations at a time (thus forming an imaginary cube). If all eight vertices of the cube have zero status, or one status, then the cube is entirely above or entirely below the surface and no faces are emitted. Otherwise, the cube straddles the surface and some faces and vertices are generated (note that as a mesh vertex we will take all vertices of generated faces). Since each vertex can have zero or one status, there are technically 2^8 possible configurations, but many of these are equivalent to one another. There are only 15 unique cases, shown in figure 5.

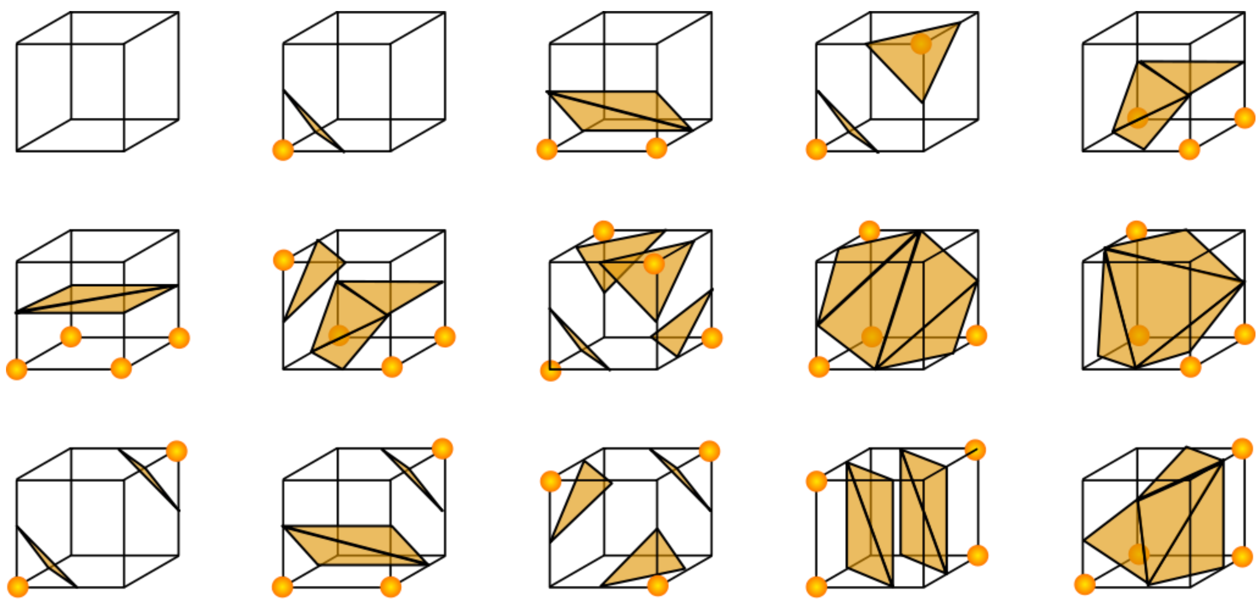

Figure 5: The originally published 15 cube configurations.

We iterate over all cubes, adding triangles to a list, and the final mesh is the union of all these triangles and their vertices.

For extracting the 3D mesh corresponding to a new observation given a trained occupancy network $f_\theta(p, x)$, we first discretize the volumetric space at an initial resolution and evaluate the network for all p in this grid. We mark all grid points p as occupied for which $f_\theta(p, x)$ is bigger or equal to some threshold τ. Next, we mark a grid point as active if at least two adjacent grid points have differing occupancy predictions. Then, using the marching cubes algorithm, we obtain the 3D mesh.

Photo Wake-Up

The key technical idea of this method is how to recover an animatable, textured 3D mesh from a single photo. At first, It recovers the SMPL mesh by estimating shape and pose parameters using the Keep it SMPL[2] method. Then the SMPL silhouette is warped to match the person silhouette in the original image, that warp is applied to projected SMPL normal maps and skinning maps as well. The resulting normal and skinning maps can be constructed for both front and back views and then lifted into 3D, along with the fitted 3D skeleton, to recover a rigged body mesh that exactly agrees with the silhouettes, ready for animation. The center box in *figure 6* illustrates this approach.

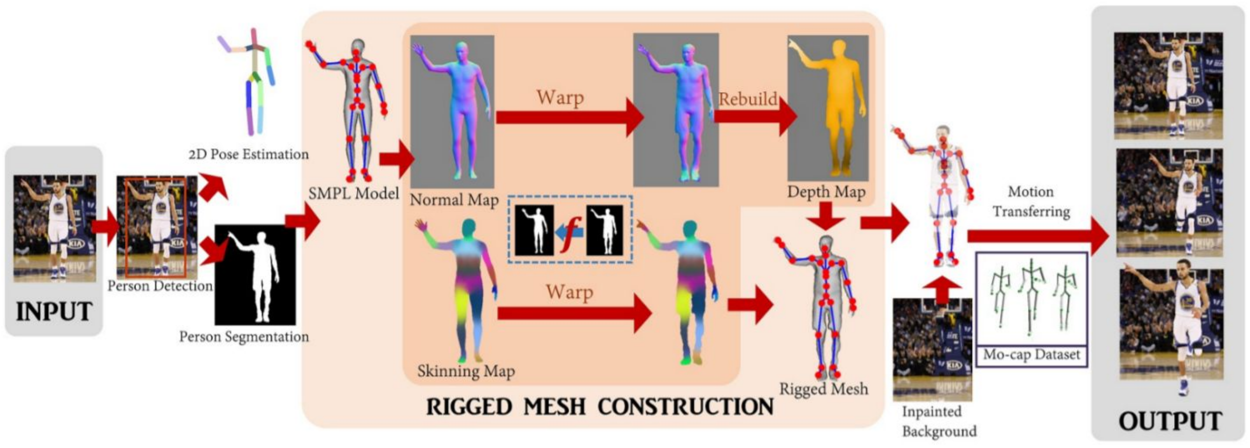

Figure 6: Given an image, person detection, 2D pose estimation, and person segmentation methods are used using existing methods. Then the SMPL model is fit to the 2D pose and projected into the image as a normal map and a skinning map. The method tries to find a mapping between a person's silhouette and the SMPL silhouette, warp the SMPL normal/skinning maps to the output, and build a depth map by integrating the warped normal map. This process is repeated to simulate the model's back view and combine depth and skinning maps to create a complete, rigged 3D mesh. Then mesh can move using any motion capture sequences.

For front texture, this method projects the image onto the geometry, back texture is an open research problem. They provide two options:

- paste a mirrored copy of the front texture onto the back, and
- inpaint with optional user guidance.

Both of these methods do not recover the back texture in a reasonable way.

3. METHODS

Given a single image, our goal is to reconstruct the underlying 3D fully textured and rigged mesh of the human body. The overall system works as shown in *figure 7*: At first, we apply state-of-the-art algorithms to perform person segmentation and image inpainting. Then we use human mesh recovery[5] method to obtain shape and pose parameters for the SMPL model. For textured mesh reconstruction we use pixel aligned implicit function[7]. Human mesh recovery and pixel aligned implicit function will be described in detail in the following subparagraphs. Finally, obtained two meshes are aligned to get skeleton and blend weight for textured mesh. We also experimented with the online service MIXAMO, the results are almost the same for images without self-occlusions.

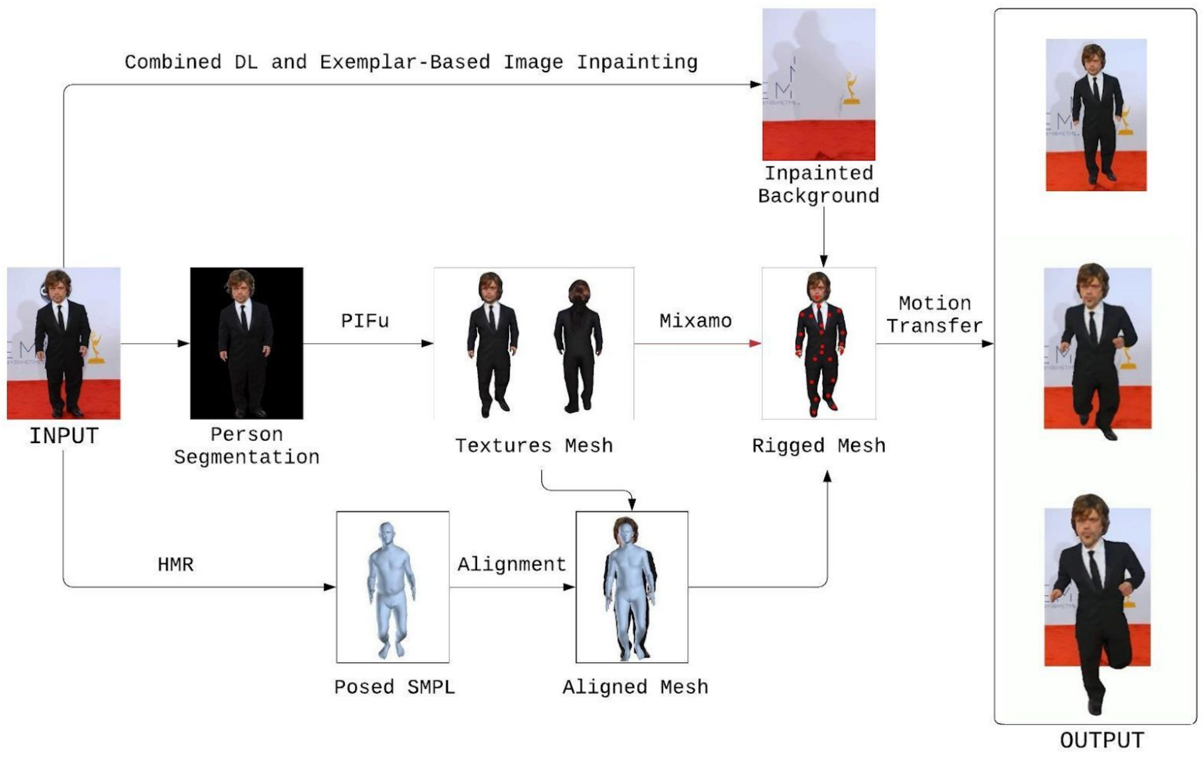

Figure 7: Overview of our method.

Unlike existing methods, which give partially textured meshes or synthesize the back regions based on frontal views, our approach, owing to pixel aligned implicit function, gives fully textured mesh.

The code is available at: https://github.com/lev1khachatryan/Tex-An_Mesh repository.

The method consists of three stages:

1. SMPL mesh reconstruction
2. Textured mesh reconstruction
3. Mesh alignment and rigging

SMPL Mesh Reconstruction

Human mesh recovery[5] is a method for predicting the shape and pose parameters for the SMPL model, from a single image. We will denote this method by HMR. From shape and pose parameters we can reconstruct the SMPL mesh.

The overall HMR method work is shown in *figure 8.*

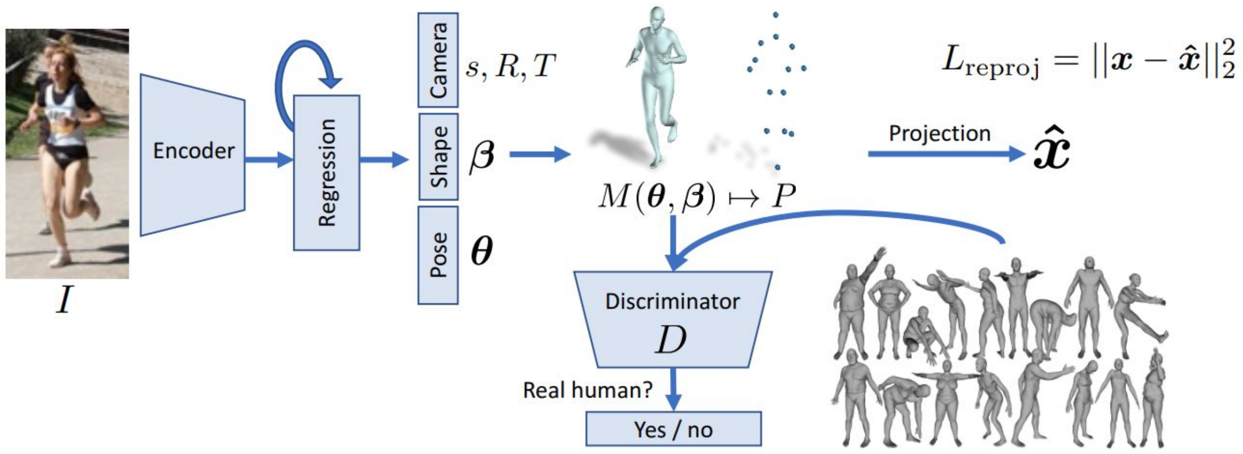

Figure 8: Overview of HMR method. At first, the image is passed through an encoder which is ResNet-50, the encoded image is used by an Iterative regression module that infers the 3D representation by minimizing the joint reprojection error. Discriminator controls the regression module to predict real human shape and pose parameters.

During training, this method assumes that we have ground truth 2D joints $x_i \in R^{2 \times K}$. Then joint reprojection error will be:

$$L_{2D\ reproj} = \sum_i \|x_i - \hat{x}_i\|_1$$

Additional, if we have paired ground truth 3D data, then 3D supervision may be employed:

$$L_{3D} = L_{3D\ joints} + L_{3D\ smpl}$$

$$L_{3D\ joints} = \|X_i - \hat{X}_i\|_2$$

$$L_{3D\ smpl} = \|\ [\beta_i,\ \theta_i] - [\hat{\beta}_i - \hat{\theta}_i]\ \|_2$$

Furthermore, a discriminator network is used to tell whether SMPL parameters correspond to a real body or not.

Textured Mesh Reconstruction

Pixel-aligned implicit function[7] is a method for reconstructing a textured mesh, from a single image. We will denote this method by PIFu. It consists of two functions: f_V and f_C, where f_V is used for 3D occupancy field reconstruction (from which we will obtain a 3D mesh using marching cubes algorithm), and f_C is used for predicting the color of each vertex of the obtained mesh. The overall inference procedure is shown in *figure 9.*

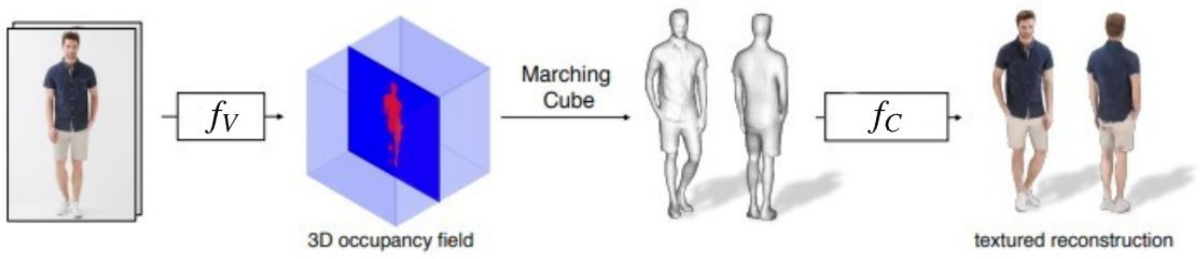

Figure 9: Given an input image, for discretized $[-a,\ a] \times [-a,\ a] \times [-a,\ a] \in R^3$ points f_V predicts the continuous inside/outside probability field. Similarly, f_C for texture inference, infers an RGB value at a given 3D point.

f_V and f_C functions are approximated with neural networks. f_V trained by minimizing the average of mean squared error:

$$L_V = 1/n \sum_{i=1}^{n} |f_V(F_V(x_i),\ z(X_i)) - f_V{}^{*}(X_i)|^2$$

where $X_i \in R^3$, $F_V(x) = g(I(x))$, (g is an image encoder and x is the 2D projection of X) and f^* is the ground truth surface. To train f_C, we need to make some changes in the loss function to avoid overfitting:

$$L_C = 1/n \sum_{i=1}^{n} |f_C(F_C(x_i{}',\ F_V),\ z(X_i{}')) - C(X_i)|$$

where $C(X_i)$ is the ground truth RGB value on the surface point X_i, $X_i{}' = X_i + \varepsilon * N_i$ and $\varepsilon \sim N(0,\ 1)$. So the image encoder for F_C has image features learned for F_V, which helps the encoder to focus on color inference only. The overall training procedure is shown in *figure 10.*

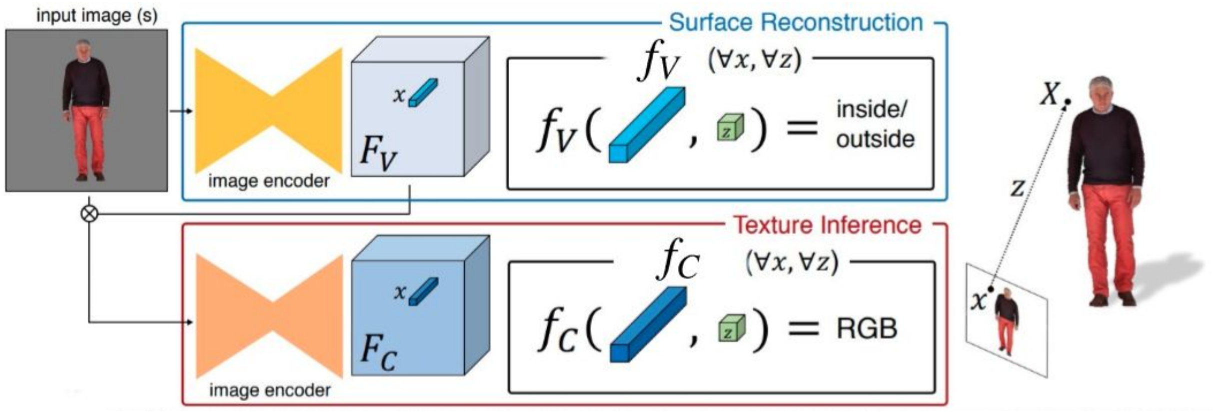

Figure 10: Overview of training procedure.

Any fully convolutional neural network can be used as an image encoder. f_V and f_C are based on MLP, where the number of neurons are (257, 1024, 512, 256, 128, 1) and (513, 1024, 512, 256, 128, 3).

Mesh alignment and rigging

We propose a new method for mesh rigging, that is based on SMPL mesh. Method is called a SMPL alignment and fitting. Moreover, we experimented with the online service MIXAMO for auto-rigging.

The method assumes that we have a not rigged mesh M_1, and SMPL mesh M_2. As we have mentioned in the related work, for the SMPL model we have the underlying skeleton, blend weights, T-pose mesh and other parameters. So, if we align vertices of M_1 to vertices of M_2 in a reasonable way, then we can take all necessary parameters to make M_1 animatable.

For each mesh, we find the minimum bounding box (cube) containing it by simply tracking the max/min for each X, Y and Z axes. Using these values we find the center of the mesh as well as the extents. The width is the distance between X_{min} and X_{max}, the depth and height are similarly calculated depending on which axis we are using for which dimension. Then we center the mesh on the origin. Finally, we scale M_1 by the size of M_2 and obtain results shown in *figure 11 (4).*

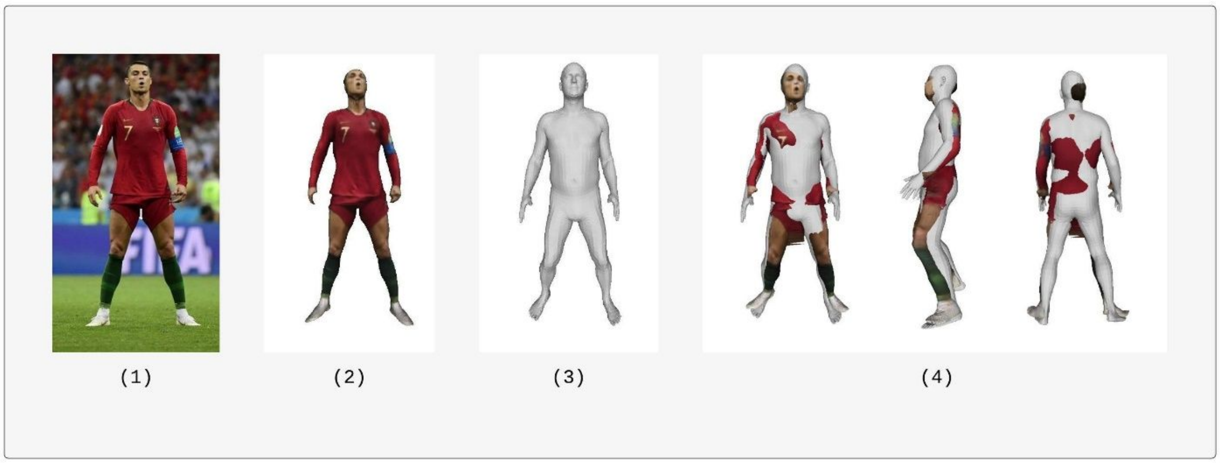

Figure 11: The result of smpl alignment.

After alignment, for each vertex of M_1 we find the nearest vertex from M_2 (distance is defined by the Euclidean metric), and take its blend weights. As a joint locations for M_1, we just copy all joint locations from M_2. In this point we have a rigged M_1 mesh, but as we use linear blend skinning for animation, then additionally we need to find the M_1 mesh in the rest pose. To that end, we need to notice, that M_2 is obtained from SMPL rest pose mesh by applying pose-dependent transformations to each joint (remember that HMR outputs β and $\theta = [\theta_1, \theta_2, \ldots, \theta_K]$, where θ_k is the axis-angle representation of the relative rotation of joint k with respect to its parent in the kinematic tree), therefore to find the rest pose M_2 mesh, we just need to apply inverse transformations to each joint of M_2, and since we have aligned M_1 to M_2, we can apply these inverse transformations to M_1 and as a result, obtain M_1 in the rest pose. You can find the results in *Figure 12.*

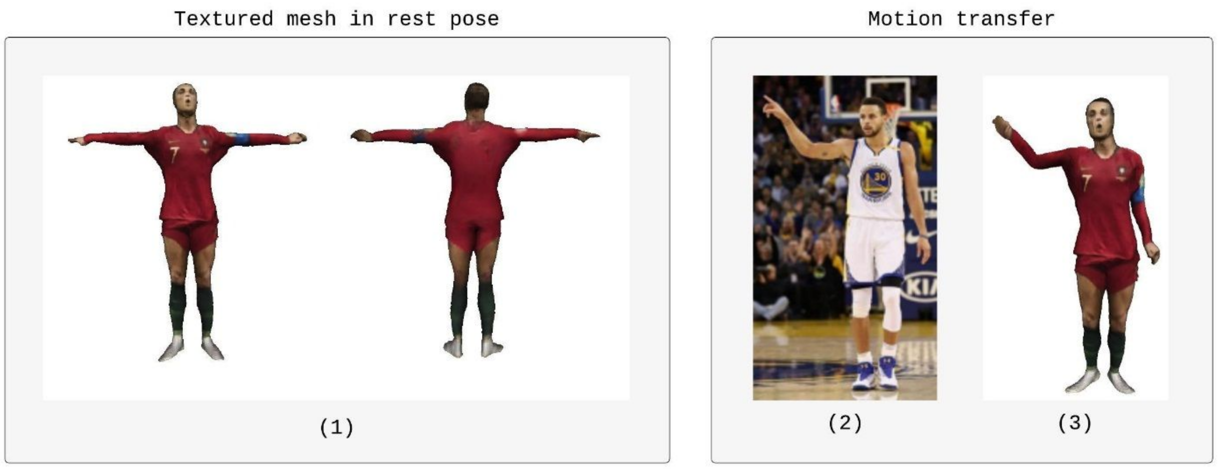

Figure 12: Textured mesh in the rest pose (1). (3) is a result of applying LBS on textured and rigged mesh.

For images with self-occlusion, our method does not work well, therefore we experimented with the online rigging service MIXAMO. It applies machine learning to understand where the limbs of a 3D mesh are and to insert a skeleton into the 3D mesh. This is not a fully automatic method, as the user must first place markers on key points (wrists, elbows, knees and groin). Besides that, we need to ensure that the mesh belongs to a humanoid with distinguishable head, body, arm, and leg areas. If the mesh proportions are too deformed, the MIXAMO may not work.

4. RESULTS

Below we present the visual results of our method. We have tested the method on 10 photos downloaded from the Internet (actors, football players and presidents that satisfied our photo specifications-full body, mostly frontal). Our simple animation tool allows us to change the viewpoint during animation and saves the result to the video. *Figure 9* and *10* show our typical animation results.

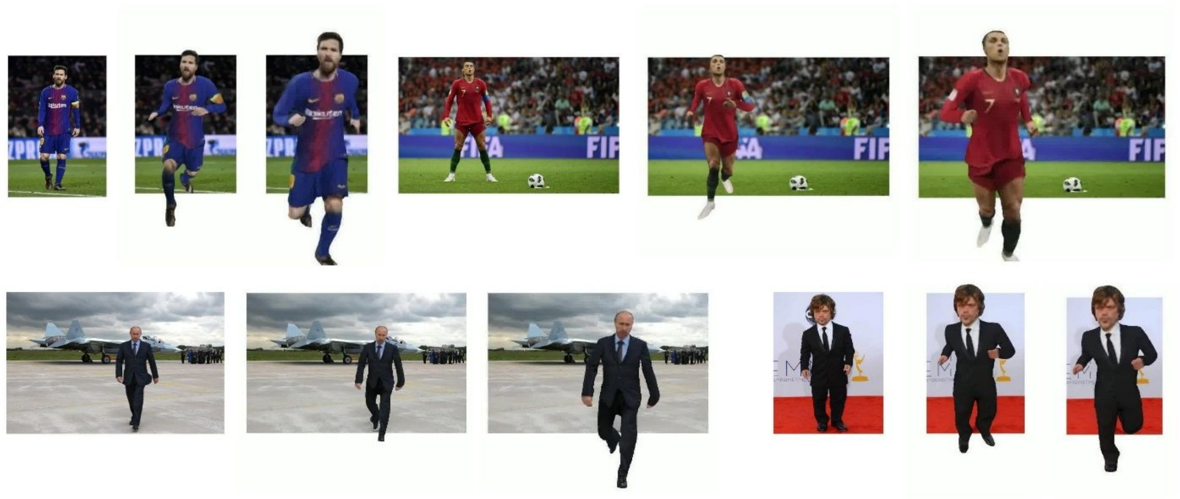

Figure 13: Characters walk and run animation results. The input is always on the left.

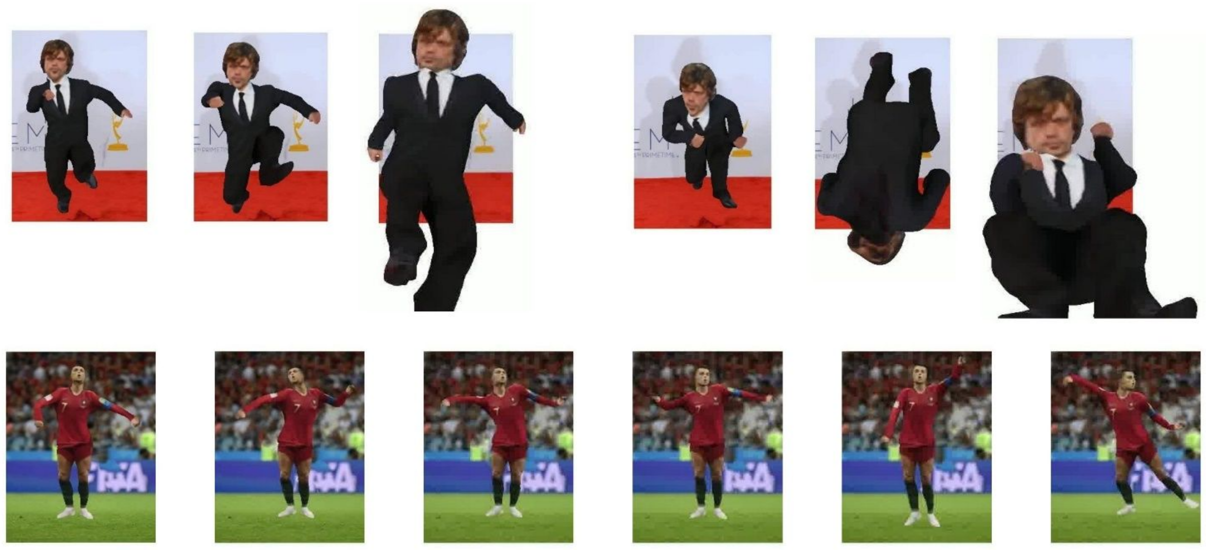

Figure 14: Animation of jump, roll and dance.

5. FUTURE WORK

In all our examples, none of the segmented subjects are occluded by any other objects or scene elements. In real-world settings, occlusions often occur and perhaps only a part of the body is framed in the camera. Being able to predict complete mesh in partially visible settings could be highly valuable for analyzing humans in unconstrained settings.

We use mesh alignment for rigging, which is not the best solution as it depends on SMPL shape-pose parameters prediction. Using machine learning to rig the mesh without the SMPL model could improve results and make the method one stage.

6. CONCLUSION

In this project, we addressed the problem of reconstructing a fully textured and animatable human body mesh from only a single image. Unlike existing methods, which give partially textured meshes or synthesize the back regions based on frontal views, our approach, owing to PIFu, gives fully textured mesh.

Our contributions are the following:

- we obtain a fully textured and animatable mesh by combining textured mesh and SMPL predictions,
- to the best of our knowledge, this is the first work which makes the predicted fully textured human body mesh animatable.

References

[1] Adobe Inc. https://www.mixamo.com/

[2] Bogo F., Kanazawa A., Lassner C., Gehler P., Romero J. and J Black M. . *Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image*. In European Conference on Computer Vision, pages 561–578, 2016.

[3] Loper M., Mahmood N., Romero J., Pons-Moll G. and J.Black M. . *SMPL: A skinned multi-person linear model.* ACM Transactions on Graphics, 34(6):248, 2015.

[4] Kavan L. . *Skinning: Real-time Shape Deformation - Direct Skinning Methods and Deformation Primitives*. Siggraph Course, 2014

[5] Kanazawa A., J Black M., Jacobs D., Malik J. *End-to-end Recovery of Human Shape and Pose*. In IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[6] Mescheder L., Oechsle M., Niemeyer M., Nowozin S. and Geiger A. *Occupancy networks: Learning 3d reconstruction in function space*. In European Conference on Computer Vision, 2018.

[7] Saito S., Huang Z., Natsume R., Morishima S., Kanazawa A. and Li H. *Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization*. In ICCV, 2019.

[8] Weng C., Curless B., Kemelmacher-Schlizerman I. . *Photo Wake-Up: 3D Character Animation from a Single Photo.* In IEEE Conference on Computer Vision and Pattern Recognition, 2018.

[9] William E. Lorensen, Harvey E. Cline. *Marching Cubes: A High Resolution 3D Surface Construction Algorithm.* 1987.