# Project Final Report

## "Group 15: OpenPose & SMPL"

Michael Stark, Daniil Zauzolkov, Jonas Helms, Yu Chen

## 1 Introduction

Recreating a human 3D mesh from a 2D Image is a very difficult task as the information of the image is inherently incomplete for 3D reconstruction. 2D Images contain no depth information, meaning that the reconstruction of a 3D mesh is only possible if we estimate the camera pose information and restrict the human body by making certain assumptions. Furthermore, factors like occlusions and obstructing clothing create the demand to have a general 3D human body model that can be fitted to an assumed pose. Despite these difficulties, Federica et al. [1] developed the *SMPLify* system which reconstructs a 3D human pose and shape from a single RGB image based on the recently developed 3D body shape model known as *SMPL* [5]. Our project aims to first re-implement the approach proposed in [1] as a `C++` application and then try to further improve the prediction accuracy by implementing our own camera estimation and using more detailed 2D body joint locations similar to the improvements in *SMPL-X* [6].

## 2 Related Work

*SMPLify* uses *DeepCut* [7] to predict the 2D joint locations of human beings in real world images. Our project will use the *OpenPose* [2] system for this task. For the second part of the pipeline, the body model, *SMPLify* uses the *SMPL* [5] model. The *Skinned Multi Person Linear model* is a learned vertex based model that represents a wide variety of body shapes and poses by adjusting 72 pose and 10 shape parameters. These parameters are learned from a rest pose template, blend weights and pose dependant blend weights. Since we wanted to develop a `C++` application we can not use the standard *SMPL* implementation and instead use a `C++` version called *SMPLpp* from [3].There have also been improvements to *SMPLify* over the years like *SMPL-X* which also optimizes for hand and face features from which we took inspiration to improve the pipeline established in *SMPLify*.

## 3 Method

Figure 1 provides an overview over our code pipeline. In the following we will describe the mathematics behind our optimization routines.

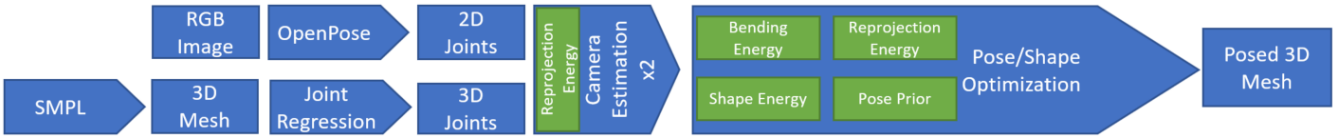


Figure 1: Pipeline Overview

### 3.1 Joint Regression

The joint regression from the 3D model is an important part of the pipeline as it makes the joint projection in the energy terms possible. The joint regression in *SMPL* specifies a specific vertex for each joint of the model represented by the 3D location in object space. The problem with this approach is that a vertex is always located on the surface of the model instead of inside, like a real joint. For this reason we handpicked a set of vertices representing each joint $j$ and average their position to conclude the final 3D joint location, which will be referred to as $F_j$ in the energy terms.

### 3.2 Camera Estimation

To reproject the joints of the 3D mesh to the image space that is equivalent to the original image it is required to estimate the camera extrinsic and intrinsic matrix. The intrinsic matrix depends on the original image size in pixels, which is a known property while the extrinsic matrix represents the orientation and position of the camera in world space. Since optimizing for the values of the extrinsic matrix did not provide sufficient results we further improve the orientation of the camera by rotating it around the two body axis of the 3D model in a second optimization pass. In the first pass we optimize for the rotation $Rot_{3\times3}(\phi_x, \phi_y, \phi_z)$ and translation $T_{3\times1}(x, y, z)$ in the extrinsic matrix $C_{extr}$ by using a similar projection approach as is used in body mesh optimization (section 3.3). Instead of optimizing for all joints we just use the *central hip* and *neck* joints of the T-pose template mesh $\bar{T}$ provided by [5] because we assume that these joints are not rotating around their parent.

$$C_{extr} = \begin{bmatrix} Rot_{3\times3} & T_{3\times1} \\ 0_{1\times3} & 1 \end{bmatrix} \tag{1}$$

The rotation matrix $Rot_{3\times3}$ is dependant on the three axis-specific rotation matrices $R_z$, $R_y$ and $R_x$ which are optimized through their respective angles $\phi_x$, $\phi_y$, $\phi_z$. The translation vector $\boldsymbol{T}$ is representing the $x, y, z$ coordinates of our model center in camera space. The first pass then optimizes the energy term $E_{C_{est}}$ over angles $\phi$ and translation $\boldsymbol{T}$.

$$E_{C_{est}}(\boldsymbol{\phi}, \boldsymbol{T}) = \sum_j \sqrt{(\Pi(\boldsymbol{\phi}, \boldsymbol{T}; F_j(\bar{T})) - J_{est,j})^2} \tag{2}$$

Where $\Pi$ is the projection function from 3D to 2D that uses $C_{extr}$ and $J_{est,j}$ is the estimated joint location of joint j provided by *OpenPose*. In the second camera estimation step we first calculate two rotational axes around which we rotate the camera that uses the extrinsics from the first step. The first axis $v_y$ is representing the spine of the 3D model.

$$v_y = \frac{(J_{Neck} - J_{HipCenter})}{\|(J_{Neck} - J_{HipCenter})\|} \tag{3}$$

The second axis $v_p$ is representing a rotation around an axis orthogonal to the look vector of the camera $C_{look}$ and $v_y$.

$$v_p = v_y \times C_{look} \tag{4}$$

The second optimization then finds the best rotation angles $\alpha_y$ and $\alpha_p$ of the camera around the $v_y$ and $v_p$ axes for the energy term. To implement this we use another projection energy $E_{C_{extr,final}}$ that now considers the joints from the first estimation step, *neck* and *hip center*, as well as both *shoulder* joints. For this, the second camera estimation pass uses a matrix rotation around an arbitrary axis $v$. The axes and angles for the two rotations are $v_y$, $v_p$ and $\alpha_y$, $\alpha_p$ respectively. They are then multiplied to the original extrinsic matrix, rotating the camera around the aforementioned axis of the body mesh resulting in the matrix $C_{extr,final}$.

$$C_{extr,final} = C_{extr} * Rot_{4\times4}(\alpha_y, v_y) * Rot_{4\times4}(\alpha_p, v_p) \tag{5}$$

The energy term for the second camera estimation using projection is then defined as:

$$E_{C_{extr,final}}(\alpha_y, \alpha_p; v_y, v_p) = \sum_j \sqrt{(\Pi(\alpha_y, \alpha_p; v_y, v_p, F_j(\bar{T})) - J_{est,j})^2} \tag{6}$$

### 3.3 Body Mesh Optimization

Initially we are starting with the T-pose model that is provided by [5]. This model can be adjusted by providing body shape parameters $\beta$ and pose parameters $\theta$ to the system of [5]. Thus we optimize over those parameters to recreate a 3D mesh given the joint locations from a picture. For the optimization we minimize our energy function over the body shape and pose parameters: $\min_{\theta,\beta} E(\theta, \beta)$. $E$ can be described as follows:

$$E(\theta, \beta) = \lambda_R E_R(\theta, \beta) + \lambda_G E_G(\theta) + \lambda_B E_B(\theta) + \lambda_\beta E_\beta(\beta) \tag{7}$$

where $\lambda_R$, $\lambda_G$, $\lambda_B$ and $\lambda_\beta$ are weights and $E_R$, $E_G$, $E_B$ and $E_\beta$ are energy terms.

To check whether the joint locations of the resulting mesh are consistent with the joint locations retrieved from *OpenPose* we project our 3D joint locations to 2D and optimize over the distances between the backprojected joint locations and the retrieved joint locations. This reprojection energy term $E_R$ is defined as:

$$E_R(\theta, \beta) = \sum_j w_j \sqrt{(\Pi(F_j(M(\beta, \theta; \Phi))) - J_{est,j})^2} \tag{8}$$

where $w_j$ are the confidence values for each joint provided by *OpenPose*, $\Pi$ is the reprojection function that takes the 3D joint location and produces 2D pixel coordinates using the optimized extrinsics described in section 3.2, $M(\beta, \theta; \Phi)$ is the body mesh created by the model from [5]. This approach is similar to the one used in [1].
We reuse the pose prior from [1] to verify that the provided $\theta$ values describe a realistic human pose in our second energy term $E_G$:

$$E_G(\theta) = -\log(\sum_i g_i \mathcal{N}(\theta; \mu_{\theta,j}, \Sigma_{\theta,i})) \tag{9}$$

Our third energy term $E_B$ is responsible for penalizing unnaturally bent knees and elbows equally to the approach in [1]:

$$E_B(\theta) = \sum_i \exp \theta_i \tag{10}$$

Our last energy term $E_\beta$ is responsible to limit the $\beta$ values in growing too large since that can cause unnatural body deformations:

$$E_\beta(\beta) = \sum_i \beta_i^2 \tag{11}$$

# 4 Evaluation

## 4.1 3D Mesh Comparison

Firstly, we demonstrate four 3D meshes and two outliers generated by our system corresponding to the RGB images from the *LSP* dataset[4], and compare our 3D meshes with the ones generated by the *SMPLify*[1] system. The rendered images are listed in Figure 2 and Figure 3. We found that even though some outlier 3D meshes e.g. **e** and **f** may appear due to the restriction of joint estimation using *OpenPose*, our system is able to reconstruct 3D body meshes capturing both the shape and pose of the human body from the 2D images. In comparison with *SMPLify* our meshes show a general better head orientation, thanks to our pipeline also considering joints in the face area. We should note that our optimization was limited in the number of iterations because of runtime issues. We found that increasing the number of iterations significantly improves the final accuracy.

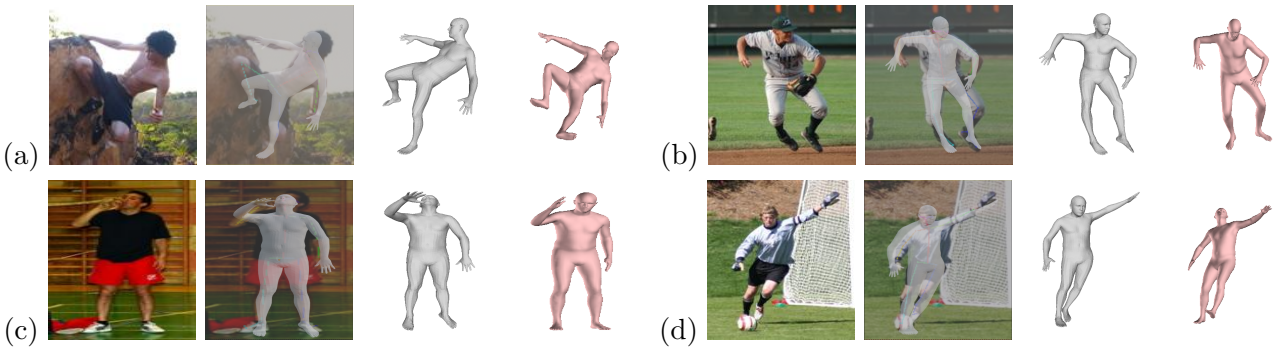


Figure 2: LSP RGB image, image fits with our mesh, our 3D mesh(white), and *SMPLify* 3D mesh(pink)

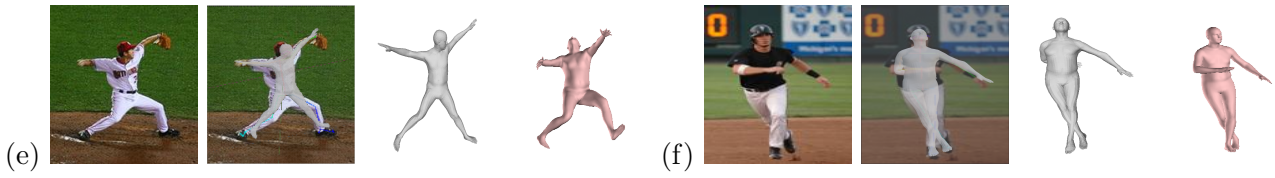


Figure 3: Outliers: LSP RGB image, image fits with our mesh, our 3D mesh(white), and *SMPLify* 3D mesh(pink)

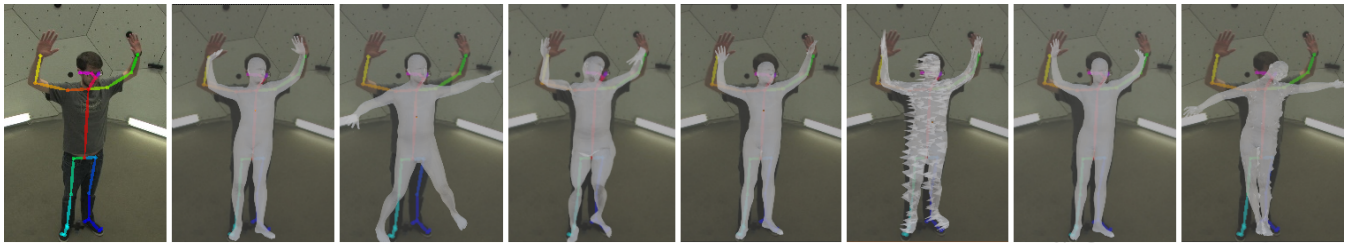


Figure 4: Ablation 3D Meshes

### 4.2 Ablation Study

In section 3, we have introduced four energy terms $E_R$, $E_G$, $E_B$, and $E_\beta$ to produce reasonable body shapes and poses using *SMPL*. Additionally, we implemented two energy terms $E_{C_{est}}$ and $E_{C_{extr,final}}$ for the camera estimation. In the following we show the impact of these energy terms on the final result: according to Figure 4, we can see that the $E_R$ and $E_{C_{est}}$ are decisive for the 3D model pose, since they highly influence to what extent the projected 2D joints fit the estimated joints. Without $E_R$ and $E_{C_{est}}$, the final 3D meshes (**c** and **h** in Figure 4) loose the pose information from the input image. From the perspective of the model shape parameter $\beta$, $E_\beta$, and $E_G$ influence it the most for this image. Figure **f** shows that the model shape contains noise without $E_\beta$. Figure **d** points out that the final mesh suffers realism without $E_G$. Finally Table 1 shows that $E_{C_{est}}$ highly influences the convergence of the optimization.

Table 1: Cost Evaluation

| | $E$ | $E$-$E_R$ | $E$-$E_G$ | $E$-$E_B$ | $E$-$E_\beta$ | no $E_{C_{extr,final}}$ | no $E_{C_{extr}}$ |
|---|---|---|---|---|---|---|---|
| Initial | 2.768163e+04 | 2.139919e+04 | 8.053853e+03 | 2.061208e+04 | 2.102947e+04 | 2.931781e+04 | **9.217143e+08** |
| Final | 1.285011e+02 | 5.661733e-01 | 1.688792e+02 | 1.249184e+02 | 1.072133e+02 | 1.509291e+02 | **5.112742e+08** |

## 5 Conclusion

Our approach has shown promising results in comparison to the results from *SMPLify* at which we were aiming. In many pictures we came close to the "ground truth" pose and shape. In several cases our meshes look as realistic or even more realistic, then the ones created by *SMPLify*. Yet there are multiple ways how our system can be augmented: to achieve more precision and realism of the optimized mesh, the iteration number of our optimizer can be increased. For a faster optimization, the whole routine can be moved to a GPU. The current optimization runs on 2D key point comparison, which can't provide information about the limb occlusion order. To avoid meshes, where limbs are occluded differently than in the input picture, image segmentation techniques could be used to provide additional information to the optimizer.

# References


[1] Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. Keep it smpl: Automatic estimation of 3d human pose and shape from a single image, 2016.

[2] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[3] Yuki Koyama Chongyi Zheng. Smplpp, 2021.

[4] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *bmvc*, volume 2, page 5. Citeseer, 2010.

[5] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 34(6):248:1–248:16, October 2015.

[6] Georgios Pavlakos, Vasileios Choutas, Nima Ghorbani, Timo Bolkart, Ahmed A. A. Osman, Dimitrios Tzionas, and Michael J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 10975–10985, 2019.

[7] Leonid Pishchulin, Eldar Insafutdinov, Siyu Tang, Bjoern Andres, Mykhaylo Andriluka, Peter V. Gehler, and Bernt Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.