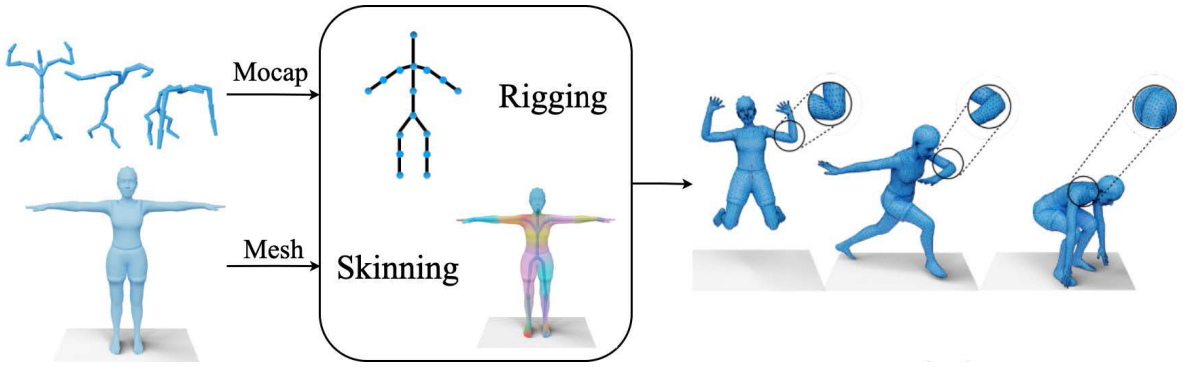
Motion capture data decorated with SMPL

121090003 Bao Jingzhi

CUHK(SZ)

日期：2022 年 8 月 17 日



1 开发环境

IDE：Microsoft Visual Studio Code (Universal)

Python：3.10.4

NumPy：1.21.2

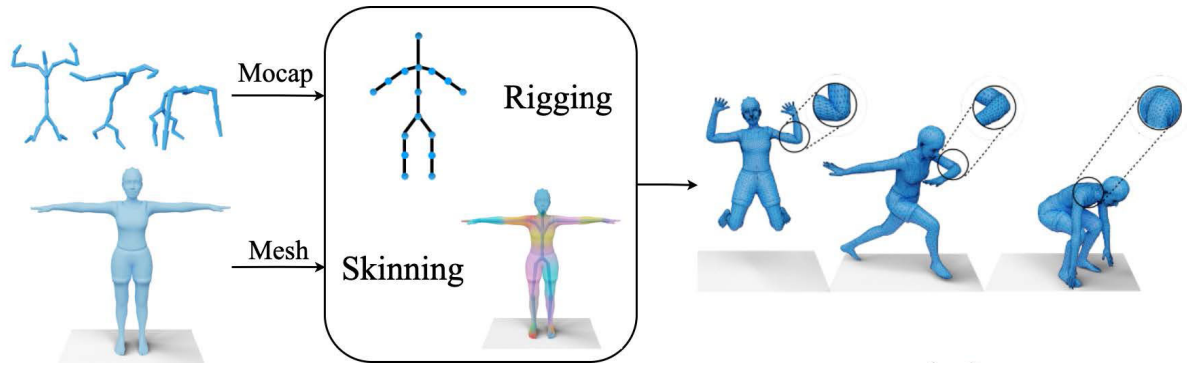
Pygame：2.1.2

transform3d：0.3.1

PyOpenGL：3.1.6

pyPickle：1.1.0

2 问题描述



我们需要根据 CMU MoCap 动作捕捉数据集重建人体动作动画，利用 SMPL 完成骨骼绑定、蒙皮等目标。

3 SMPL 模型

SMPL 模型是一种基于顶点的线性蒙皮方案，且在一定程度上解决了模型姿态形变时局部区域凹陷的失真，具有准确度高，兼容主流渲染等优点。SMPL 将人的动作理解为在一个模板模型不同关节 (joint) 处进行形变的总和。

3.1 原理简介

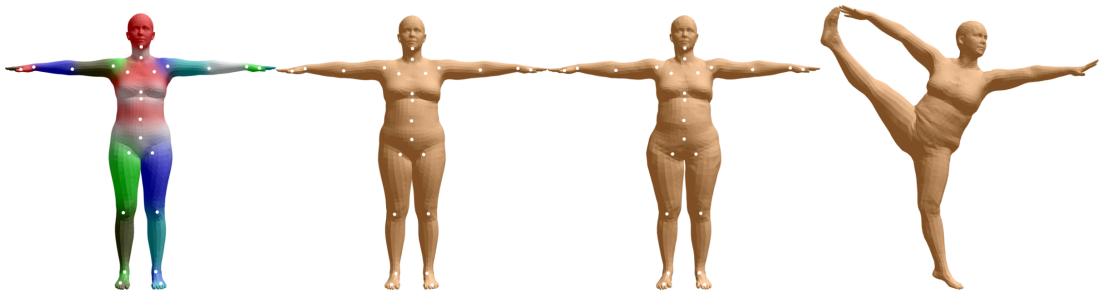


Figure 3: SMPL model. *(a) Template mesh with blend weights indicated by color and joints shown in white. (b) With identity-driven blendshape contribution only; vertex and joint locations are linear in shape vector $\vec{\beta}$. (c) With the addition of of pose blend shapes in preparation for the split pose; note the expansion of the hips. (d) Deformed vertices reposed by dual quaternion skinning for the split pose.*

SMPL 定义了一个模板模型 (a)，并通过数据集学习得到了两组参数：$\mathcal{S} = \left[\mathbf{S}_1, \ldots, \mathbf{S}_{|\vec{\beta}|}\right] \in \mathbb{R}^{3N\times|\vec{\beta}|}$，$\mathcal{P} = [\mathbf{P}_1, \ldots, \mathbf{P}_{9K}] \in \mathbb{R}^{3N\times 9K}$ 并通过改变加权值参数 β^{10} 和 $\theta^{24\times 3}$ 分别控制初始状态人体的体态 (body shape) 和姿态 (pose shape)。更明确地，$\mathcal{S}$ 用于在模板模型的基础上改变人在标准状态下的体形特征，$\mathcal{P}$ 用于在关节旋转前修正关节附近的形态，以防止在模型作用 pose 后局部产生凹陷的现象。因此，经过这两组参数的预处理，我们得到的形变前的模型如图 (c) 所示，最后对指定关节作用 pose 就得到了图 (d) 所示的效果。

3.2 模型概述

3.2.1 混合蒙皮 (Blend skinning)

如上文所述，SMPL 将人的动作理解为在一个模板模型不同关节处进行形变的总和，我们用 $\vec{\omega}_k \in \mathbb{R}^3$ 表示关节点的旋转向量。模型的 $K = 23$ 个关节点和一个 root 节点构成了一个 pose $\vec{\theta}^{24\times 3} = \left[\vec{\omega}_0^T, \ldots, \vec{\omega}_K^T\right]^T$。在笛卡尔坐标系下，旋转向量对顶点的作用需要转化为 Rodrigues 矩阵

$$\exp\left(\vec{\omega}_j\right) = \mathcal{I} + \widehat{\omega}_j \sin\left(\|\vec{\omega}_j\|\right) + \widehat{\omega}_j^2 \cos\left(\|\vec{\omega}_j\|\right) \tag{1}$$

其中

$$\vec{\omega} = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \tag{2}$$

为优化模型旋转后的表现，作者将顶点 (vertex) $\bar{\mathbf{t}}_i$ 旋转后的坐标定义为多个 ($\leq$ 4) 骨骼旋转作用的加权和，$\mathcal{W} \in \mathbb{R}^{N\times K}$。结合关节点的初始位置 $\mathbf{J}$，可以写出所有顶点的坐标变换公式。SMPL 模型的主要任务是计算如下的变换矩阵 G，再对全体顶点应用变换 G 得到目标状态

$$\begin{aligned} \bar{\mathbf{t}}_i' &= \sum_{k=1}^{K} w_{k,i} G_k'(\vec{\theta}, \mathbf{J}) \bar{\mathbf{t}}_i \\ G_k'(\vec{\theta}, \mathbf{J}) &= G_k(\vec{\theta}, \mathbf{J}) G_k\left(\vec{\theta}^*, \mathbf{J}\right)^{-1} \\ G_k(\vec{\theta}, \mathbf{J}) &= \prod_{j\in A(k)} \left[\begin{array}{c|c} \exp\left(\vec{\omega}_j\right) & \mathbf{j}_j \\ \hline \overrightarrow{0} & 1 \end{array}\right] \end{aligned} \tag{3}$$

3.2.2 体态混合 (Shape blend shapes)

为初始化体态，模型中提供了由机器学习得到的参数 $\mathcal{S} = \left[\mathbf{S}_1, \ldots, \mathbf{S}_{|\vec{\beta}|}\right] \in \mathbb{R}^{3N\times|\vec{\beta}|}$ 和一组由用户调节的加权值 β[10]。初始体态的偏移量 (displacement) 定义为

$$B_S(\vec{\beta}; \mathcal{S}) = \sum_{n=1}^{|\vec{\beta}|} \beta_n \mathbf{S}_n \tag{4}$$

3.2.3 姿态混合 (Pose blend shapes)

$$B_P(\vec{\theta}; \mathcal{P}) = \sum_{n=1}^{9K} \left(R_n(\vec{\theta}) - R_n\left(\vec{\theta}^*\right)\right) \mathbf{P}_n \tag{5}$$

3.2.4 关节位置求解器 (Joint regressor)

作者在 pkl 文件中提供了 $\mathcal{J}$ (Joint regressor)，免去了手动标点的过程，这个参数同样由机器学习得到。将 $\mathcal{J}$ 作用于 2.2.2 中得到的模型即可得到所有关节的位置。

3.3 算法描述

经过上述铺垫，SMPL 模型的操纵可以总结为如下三步：

1. 输入参数 β, γ
2. 根据参数调整模板 (T-pose) 模型至初始状态 (rest pose)

$$T_P(\vec{\beta}, \vec{\theta}) = \overline{\mathbf{T}} + B_S(\vec{\beta}) + B_P(\vec{\theta}) \tag{6}$$

3. 计算旋转矩阵 G，更新全体顶点坐标

$$\mathbf{t}'_i = \sum_{k=1}^{K} w_{k,i} G'_k(\vec{\theta}, J(\vec{\beta}; \mathcal{J}, \overline{\mathbf{T}}, \mathcal{S})) \mathbf{t}_{P,i}(\vec{\beta}, \vec{\theta}; \overline{\mathbf{T}}, \mathcal{S}, \mathcal{P}) \tag{7}$$

3.4 SMPL 模型参数表

Attributes	Explanation	Type
pose	θ	np.array(24, 3)
beta	β	np.array(10)
trans	世界坐标下模型的平移量	np.array(3)
verts	模型所有节点的坐标	np.array(6890, 3)
parent	joints 的父节点	dict{int: int}
v_template	模板模型的全体顶点坐标	np.array(6890, 3)
J_regressor	模型 joints 的 Regressor	
J	joints 的世界坐标	np.array(24, 3)
R	pose θ 的 Rodrigues 旋转矩阵形式	np.array(3, 3)
set_params(pose, beta, trans)	设置参数信息	*function*
compute_R_G()	根据 θ 计算所有关节的 G 旋转矩阵	*function*→np.array(24, 4, 4)
do_skinning(G)	根据 G 对全体顶点作用 pose	*function*
update()	计算 G，对全体顶点作用 G	*function*
rodrigues(r)	将旋转向量 r 转化为 Rodrigues 旋转矩阵	*function*→np.array(3, 3)

4 MoCap dataset

MoCap 数据集由 ASF (base pose file) 和 AMC (motion data file) 构成，是一种基于骨骼 (bone) 的动作状态数据，其公司提供了数据解析的办法 Acclaim ASF/AMC。该数据的采集由多位点相机记录，因此文件中记录的数据采用世界坐标系。而上文提到 SMPL 模型是根据关节点定义的，因此我们还需要将骨骼数据转换成关节数据，我们采用如下的主要步骤进行数据转换 ASF data→ASF Joint Model→ 校正 SMPL Template model→SMPL Joint Model，整个过程封装在 imitator 类。下表给出了每个类需要维护的信息。

4.1 ASF Joint 参数表

Attributes	Explanation	Type
name	关节名称	string
direction	T-pose 状态下骨骼方向的正则化向量	np.array(3)
length	骨骼长度	scalar
axis	初始状态下骨骼在世界坐标下的旋转向量	list[float]
C	axis 的 Rodrigues 旋转矩阵形式	np.array(3, 3)
Cinv	C 的逆矩阵	np.array(3, 3)
parent	父节点	Joint
children	子节点	list[Joint]
coordinate	世界坐标	np.array(3)
matrix	世界坐标下的旋转信息总和	np.matrix(3, 3)
relative_R	动作变换 C×(motion rotation)×Cinv	np.array(3, 3)
set_motion(motion)	根据 motion 计算模型的 coordinate 和 matrix	*function*
to_dict()	得到整个模型的信息	*function* → dict{name: Joint}
reset_pose()	重置模型得到 T-pose 状态	*function*

4.2 SMPL Joint 参数表

Attributes	Explanation	Type
idx	SMPL 模型的关节编号	int
to_parent	父节点到此节点的向量	np.array(3, 3)
parent	父节点	SMPLJoints
coordinate	世界坐标	np.array(3)
children	子节点	list[SMPLJoints]
align_R	SMPL T-pose 到 asf 模型的纠正量	np.array(3, 3)
motion_R	由 amc 文件计算得到的旋转量	np.array(3, 3)
init_bone()	计算初始状态的 to_parent	*function*
set_motion_R(motion)	根据 motion 计算所有节点的旋转矩阵	*function*
update_coord()	更新所有节点的 coordinate	*function*
to_dict()	得到整个模型的信息	*function* → dict{idx: SMPLJoints}
export_G()	根据 motion_R 和 align_R 计算旋转矩阵 G	*function*→np.array(3, 3)

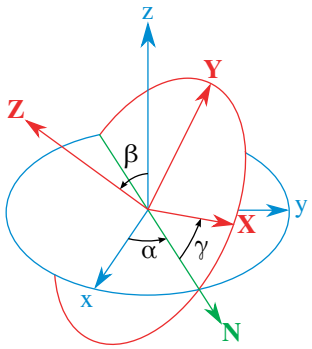
4.3 imitator 参数表

Attributes	Explanation	Type
asf_joints	asf 文件解析得到的关节点信息	dict{name: joint}
smpl	pkl 文件解析得到的 SMPL 模型	SMPLModel
setup_smpl_joints()	初始化 SMPL 模型节点	*function*→ dict{idx: SMPLJoints}
align_smpl_asf()	校准 asf 模型到 SMPL (align_R)	*function*
compute_rodrigues(x, y)	计算 align_R 满足 y = Rx	*function*→np.array(3, 3)
imitate(motion, translate =False)	根据 amc motion 进行模型演示	*function*
set_asf_motion(motion, translate)	同上	*function*
asf_to_smpl_joints(translation)	将 asf 模型的 pose 迁移到 SMPL 模型	*function*
map_R_asf_smpl()	将 asf 模型的旋转量转移到 SMPL 模型	*function*→(dict{idx: R}, T)

5 实现细节

这一部分将讨论具体实现时需要注意的一些细节，不同的实现方法可能面临的困难不尽相同，故这一部分仅基于论文给出的实现方式进行解释和提供优化。内容包括关节旋转时矩阵的应用顺序，AMC 数据到 SMPL 模型的适配和蒙皮过程中的 trick。

5.1 旋转矩阵的应用顺序



单位旋转矩阵的应用可以视为世界坐标系的旋转，如上图的蓝色坐标系在应用欧拉角 $<\alpha, \beta, \gamma>$ 后，坐标系视角转化为红色坐标系。因此旋转矩阵的作用顺序应为由局部到整体，即

global translation < parent joint rotation < joint rotation < rest pose direction

5.2 AMC 数据适配 SMPL 模型

AMC 文件的 `:bonedata` 的记录值是世界坐标下的骨骼参数（`axis` 为欧拉角表示），SMPL 模型的记录值是每个关节自身的相对旋转量。记 AMC 文件的 `axis` 记录值的 Rodrigues 形式为 R_r ，则转换关系为

$$R_{relative} = CR_rC^{-1} \tag{8}$$

5.3 蒙皮运算

公式 (3) 将蒙皮过程表达为先作用 $G_k\left(\vec{\theta}^*, \mathbf{J}\right)^{-1}$ 将模型从 rest pose 转换到 T-pose，再作用 $G_k(\vec{\theta}, \mathbf{J})$ 将模型从 T-pose 转换到给定参数的状态。在实际运算过程中

$$G_k(\vec{\theta}, \mathbf{J}) = \left[\begin{array}{c|c} \prod_{j \in A(k)} \exp\left(\vec{\omega}_j\right) & \mathbf{j}_k \\ \hline \overrightarrow{0} & 1 \end{array}\right] \tag{9}$$

若对顶点 v_i 作用此关节 k 的变换只需要先对 G_k 消除旋转量对 joint 位置的贡献

$$G'_k(\vec{\theta}, \mathbf{J}) = G_k(\vec{\theta}, \mathbf{J}) - G_k(\vec{\theta}, \mathbf{J}) \begin{bmatrix} \mathbf{j}_k \\ 0 \end{bmatrix} \tag{10}$$

最后直接应用 v_i 即可得到变换后的顶点 v'_i

$$\begin{bmatrix} v'_i \\ 1 \end{bmatrix} = G'_k(\vec{\theta}, \mathbf{J}) \begin{bmatrix} v_i \\ 1 \end{bmatrix} \tag{11}$$