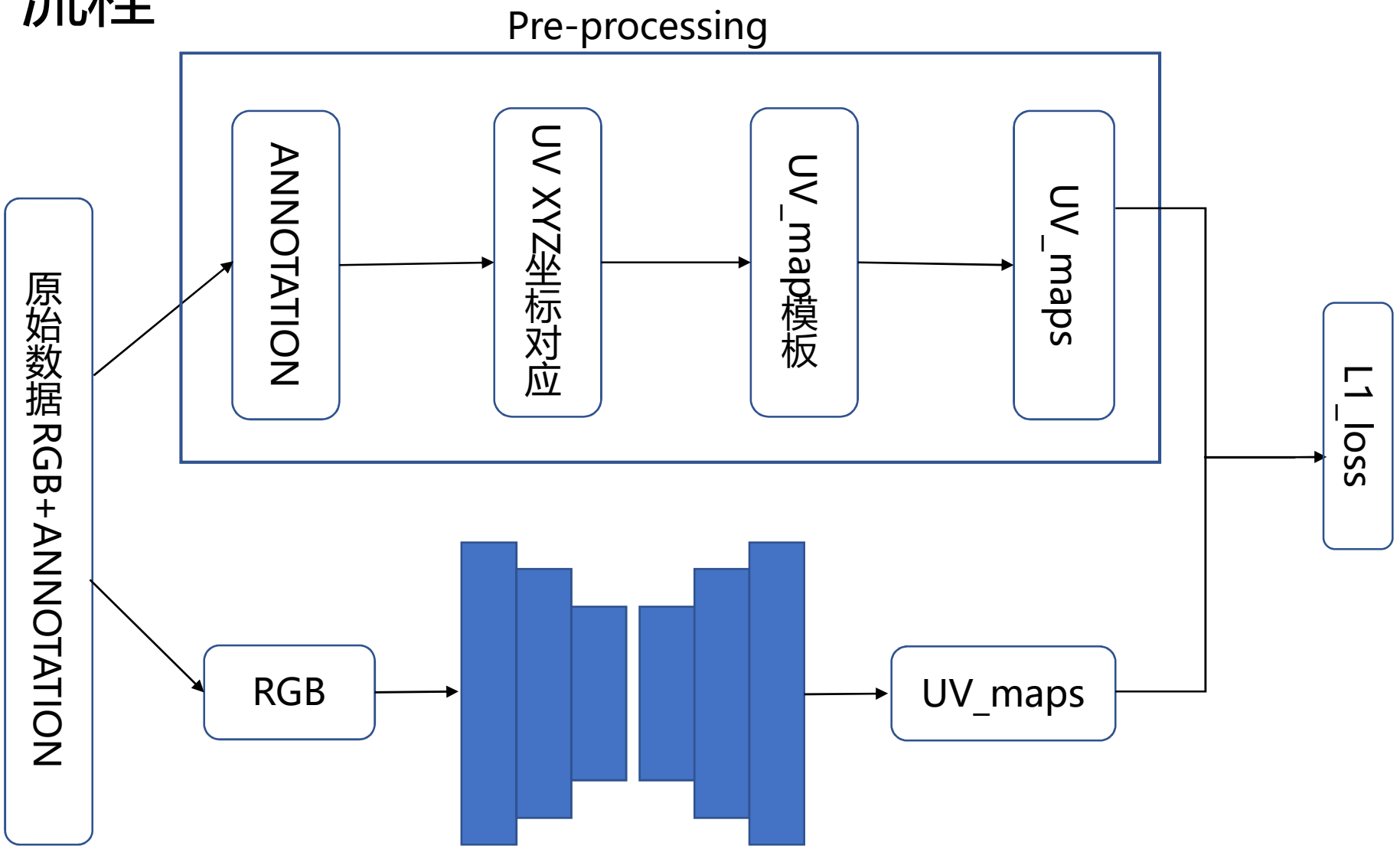
DenseBody

2019年7月22日

流程


Pre-processing
原始数据RGB+ANNOTATION
ANNOTATION
UV XYZ坐标对应
UV_map模板
UV_maps
L1_loss
RGB
UV_maps

数据处理

源数据：RGB图像和annot.h5
keys=[center, gt2d, gt3d, height, weidth, imagename, pose, shape, smpl_joint]

需要的格式: UVmap和annot_convert.npy
keys=[gt2d, gt3d, imagename, pose, shape, *mesh, *joints]
其中*表示可以通过pose和shape得到

数据处理-SMPL

- SMPL name：neutral_smpl_with_cocoplus_reg.pkl

```
keys = [
    J_regressor, # Regressor for joint locations give    # n shape - 6890 x 24
    cocoplus_regressor, # returns 19 keypoints: 6890 x 19. if lsp :joint_regressor[:, :14]
    weights,   # LBS weights
    posedirs,  # Pose blend shape basis: 6890 x 3 x 207
    v_template,  # Mean template vertices
    shapedirs,  # Shape blend shape basis: 6980 x 3 x 10
    f,
    kintree_table # indices of parents for each joints
]
```

- 有些地方cocoplus_regressor也叫joint_regressor

SMPL

- 给定thetas和beta求得mesh和joint

  首先明确theta和beta的所储存的变量类型
- **thetas:** an [N, 24 * 3] tensor indicating child joint rotation relative to parent joint. For root joint it's global orientation. Represented in a axis-angle format.
- **betas:** Parameter for model shape. A tensor of shape [N,10] as coefficients of PCA components. Only 10 components were released by SMPL author.

SMPL

- 给定thetas和beta求得mesh和joint（伪代码）

```
# 1. Add shape blend shapes
# (N x 10) x (10 x 6890*3) = N x 6890 x 3
v_shaped = beta * self.shapedirs + self.v_template

# 2. Infer shape-dependent joint locations.
J = self.J_regressor * v_shaped

# 3. Add pose blend shapes
# N x 24 x 3 x 3
# turns axis-angle tensor into rotation matrix
Rs = batch_rodrigues(theta)
# Ignore global rotation.
pose_feature = Rs - eye
v_posed = pose_feature * self.posedirs + v_shaped

# 4. Get the global joint location
# A: relative joint transformations for LBS.
# self.J_transformed, A = batch_global_rigid_transformation(Rs, J, self.parents)

# 5. Do skinning:
# W is N x 6890 x 24
T = W * A
rest_shape_h= concat([v_posed, ones([num_batch, v_posed.shape[1], 1])], 2)
v_homo = T * rest_shape_h

# get joint location
joints = joint_regressor * v_homo
```

SMPL-maya模型解析与转化

- 1. 将官方fbx模型转化成obj，obj内容解析参考
  - https://52zju.cn/?p=186887
  - # f v1/vt1/vn1 v2/vt2/vn2 v3/vt3/vn3
- 2. 根据obj内容，找到v和vt索引的对应关系，并储存
  a) 索引从1开始，不是0；
  b) v的个数为6890 （.obj文件中以‘v’开头的行），vt个数为7576（.obj文件中以‘vt’开头的行），所以v_to_vt时，有的v可能会对应多个vt，而vt_to_v则是一一对应。
  c) 共有13776个面。6900个四元数（.obj文件中以‘f’开头的行）但有的四元数只有三个顶点。分割四元数成面很随意，也可以借助SMPL model的faces进行转化, 转化过程参考 data_utils/triangulation.py
  d) 储存的keys除了v与vt索引对应关系外，还有face，以及v vt的坐标。
  e) 关于UV_map详细请看 https://github.com/Lotayou/densebody_pytorch/issues/4

生成UV_map模板

1. 读取转化maya模型后的文件，根据keys获得相应的值。
2. 因为uv坐标范围（0-1），需乘以模板图片的height，width。由于得到的坐标不一定是整数，所以引入质心插值。根据质心插值的定义

$$\begin{cases} w_1x_1 + w_2x_2 + w_3x_3 = x \\ w_1y_1 + w_2y_2 + w_3y_3 = y \\ w_1 + w_2 + w_3 = 1 \end{cases}$$

矩阵形式为

$$\begin{bmatrix} x_1 & x_2 & x_3 \\ y_1 & y_2 & y_3 \\ 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ w_3 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

注意w1 w2 w3均为正

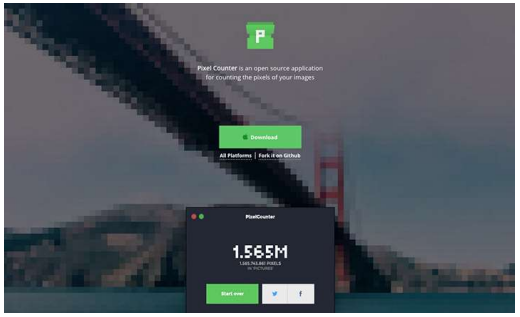
生成UV_map模板

3. 比如对图片中坐标为（x0，y0）的像素点，理论上是先找到该像素点位于哪个三角形，但是操作上比较麻烦。简单的方法是求解所有的三角形（13776个），构建shape = 13776*3*3的向量，然后w向量shape13766*3*1，（x0，y0）复制13776份shape为13776*3*1。利用numpy.linalg.solve函数求解。
4. 对于求解出来的w矩阵，如何判断是不是有效的（也就是像素点在是否在三角形内部）如果w>1e-10（阈值可以设置其他）那么认为是有效的。使用np.logical_and.reduce()函数。首先执行括号内的操作得到ture false，然后执行reduce操作，只要有一项是false，那么就舍弃。
5. 如果当前像素值不在任意三角形内，那么就将其face的索引设置为-1，如果在某个三角形内，那么将其face索引设置到当前的三角形索引，并储存w权重。
6. 对于300*300的像素而言，整个过程持续8分钟左右。（有其他质心插值方法，考虑到只需要运行一次，故暂不考虑优化）

UV_map模板优化

- 直接权重插值后，会出现边界棱角生硬的现象 Counter Pixel 形如下图，使用类似膨胀操作可以解决，同时也能解决采样的问题

膨胀前



膨胀后

UV_map模板优化

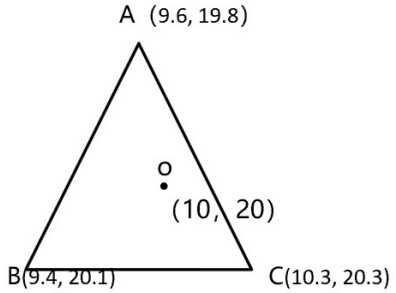
- 进行膨胀操作之前，需要记录UVmap的哪个点是边界点。
  算法操作：记录邻近8个点，如果当前像素没有对应的face，则计算邻近8个点，如果邻近8个点包括有对应face的像素点和无对应face的像素点，则说明是边界点，然后记录8个临近点中，有对应face的像素点的个数n，用于插值取样。
- 进行膨胀操作时，计算8临近点平均这（除数为上个面记录的个数n）

- 至此，可以得到优化后的UV_map

UV_map的重采样resample

1. UV_map恢复到v的坐标的过程
   8临近点。
   因为关注是UV_map的那些区域，所以索引的时候，只需要求模板图的mask>0像素点的值就行。

误差主要来源



1. resample: UV_map to verts_3D

   因为生成UV_map时，如右图所示，本来ABC三个点各自对应xyz坐标，但是由于线性插值，UV_map只会记录o点的坐标。采样的过程中，即使使用8临近点的方法，也会有一定的误差。

2. 待补充