基于比例正交投影的三维运动重建

摘要

不借助摄相机参数等先验知识，直接从单目二维视频中获取人体三维运动轨迹是一个艰难的技术挑战。前人的研究多通过将人体骨骼框架近似为可绕关键点旋转的刚体链杆组合，利用刚体中关键点之间具有固定的欧氏距离实现基于二维输入的三维运动重建。本文通过定制个性化的人体树状骨骼模型，在比例正交投影的假设下对人体二维坐标输入进行三维重建，并利用卡尔曼滤波及低通滤波对运动序列进行优化使其更符合人体在三维空间中关节运动的连续性。

首先，我们使用**三次样条插值**及**线性插值**结合置信度处理了由侦测异常、关节遮挡导致的缺失值与异常值，并利用**滑动平均**对数据进行平滑处理，绘制出坐标序列曲线图检测预处理效果。

然后，针对题目给出的人体关键点构成的人体骨骼模型，我们以关键点 1（脖子）作为根节点，建立**人体三维树状骨骼模型**，选取视频中相应链杆平行于摄影平面的若干图像帧，求平均值获取每段的相对比例长度，结合左右侧端点的数据分布，定制个性化人体模型。

接着，我们充分利用人体各段链杆长度保持不变的约束和各关节间的骨骼连接关系，构建**基于比例正交投影的三维重建模型**，分三步还原三维轨迹。第一步，我们在单帧图像中进行参数估计，估计出每帧图像中链杆在比例正交投影下的**单帧比例因子**；第二步，在根节点参考系下求解各端点的相对深度，结合人工选择与人体生物力学约束进行**单帧三维重建**；第三步，我们将坐标序列还原至地面坐标系的运动轨迹。

最后，我们利用**卡尔曼滤波**及**低通滤波**，对三维坐标序列进行 3d 优化。然后，我们利用坐标变换，求解出了以初始帧的 A 点为原点，以像素为单位的 A 点三维运动轨迹。

关键字：**三维运动重建 比例正交投影模型 卡尔曼滤波 低通滤波**

目录

一　问题重述

1.1　问题背景

基于图像的三维重建是从输入的一张或多张二维图像中恢复物体的三维几何形状的过程，是计算机视觉领域的重要研究内容之一。目前，计算机视觉的估计方法一般分为主动向目标发射光束的主动视觉法，以及通过摄像系统摄取图像作为输入的被动视觉法。在被动视觉法中，单目视觉系统具有成本低、结构简单等优点，与双目视觉和多目视觉比较，减少了图像匹配问题，且同一时间只需对单幅图像进行处理，大大降低了系统工作量及运行时间。目前，针对单目视觉的序列图像估计目标的三维运动信息的方法主要分为基于传统的视图几何的方法与深度学习方法。前者主要依靠将目标结构近似刚体，利用其上特征点之间相对固定的位置关系进行三维重建，该思路在无人驾驶、卫星定位、生物医疗、人体运动等领域具有巨大的应用价值。

近年来，基于人体运动二维输入的三维轨迹重建是人体运动姿态分析的核心内容。研究人员希望从随处可得的普通视频源中获取人体运动数据，重建人体的三维运动。在单目视觉的条件下，经过了人体的树形二维特征点选定与跟踪后，可以获得视频图像序列中人体各关节点在图像中的位置，从而得到人体的二维运动骨架序列。在此二维输入基础上，实现骨架序列的运动轨迹三维重构，成为人体运动捕捉术及姿态识别理解的关键步骤。

1.2　问题重述

在例图中，为获取机械臂末端（O 点）的三维空间运动轨迹，通常利用与其固连且不发生相对转动的手部 A 点代替 O 点作为捕捉目标，将目标转化为获取 A 点的运动轨迹。本题基于人体骨骼框架近似刚体，具有固定的欧式距离的性质，需要我们以人体骨骼的二维运动轨迹序列作为输入，对 A 点的运动轨迹进行三维重建。

本题将人体关键点对应为编号为 0-17 共 18 个骨骼特征点，并提供每个关节点在图像中的横纵坐标信息，即含有 924 帧中每一帧的位置信息的 json 文件（包括横、纵坐标及置信度三项指标）。我们需要结合关键点位置示意图及 json 文件，还原出 A 点在三维空间中的运动轨迹。

二　问题分析

本题中，已经提供了对人体骨骼框架进行运动捕捉的结果，完成了人体特征点选定与跟踪，例图与运动示例.mp4 文件也提供了模型关节点与图像特征点的对应关系，我们的目标是利用每个特征点的二维位置信息，建立特征点的三维轨迹还原模型。

首先，基于视频的人体三维运动重建通常有两种方式：采用多个摄像机或单个摄像机，本题仅给出一台摄像机在每一帧中的运动捕捉数据，属于三维重建被动估计方法中

的单目视觉输入；

其次，本题也没有提供多台摄像机对同个静止目标的数据，也没有其他先验知识（如摄像机焦距、人体与摄像机间的相对距离等），难以进行摄像机参数的自标定，也就难以求解各绝对长度。本任务是从未经定标的单目视频序列中，恢复 A 点运动的三维轨迹。因此，本题的三维重建任务可以转化为，以人体身高 H 为基准，建立基于相对长度的投影成像模型进行三维重建，将给出的图像二维运动序列还原为基于身高的三维坐标下的运动序列。

同时，已将 38.5s 的视频序列这一连续性数据转化成了 924 帧的离散型数据，要还原 A 点的运动轨迹，就需要对每一帧的二维位置还原 A 点的三维位置，再将离散数据连续化检测还原效果。

综合上述分析，本题需要我们利用 924 帧中的人体骨骼二维信息，建立三维重建模型，求解每一帧中 A 点的真实三维坐标，实现运动轨迹还原。

三　模型假设

1. 以题目示意图（图 2）中的骨骼框架为准，将人体近似成一个由 17 个关节点连接的一系列刚体的集合。

2. 每段刚体两端的人体关键点两两相对位置固定，具有固定的欧氏距离。

3. 人体各关节的相对深度和人体与摄像机之间的距离之比很小，摄像机的透视投影可近似认为是比例正交投影。

四　符号说明

表 1: 符号说明

符号	描述	单位
$J_i, i = 0, 1, \cdots, 13$	人体树状骨骼模型特征点	无
$J_i, i = 0, 1, \cdots, 12$	人体树状骨骼模型链杆相对长度	像素
$\|dZ\|$	关节之间的相对深度差	像素
(X_{ki}, Y_{ki}, Z_{ki})	人体模型端点 i 在第 k 帧中三维坐标	像素
(u_{ki}, v_{ki})	人体模型端点 i 在第 k 帧中图像平面的二维坐标	像素
s_{ki}	杆 i 在第 k 帧中的投影比例因子	无
s_k	第 k 帧确定的投影比例因子	无
dZ_{ij}	关节点 i 与关节点 j 的链杆在单帧中相对深度	像素
$direction$	相应 dZ 的符号	无
dZ_{ij}	节点 i 到节点 j 的链杆在单帧中相对深度	像素
$P_i, i = 1, 2, 3, 4$	三段杆模型中的关节点	无
$L_i, i = 1, 2, 3$	三段杆模型中的长度	无
f	摄相机焦距	像素
$Z_i, i = 0, 1, \cdots, 13$	单帧中人体模型第 i 个端点的深度	像素
x_t	卡尔曼滤波模型中 t 时刻的状态向量	无
y_t	卡尔曼滤波模型中 t 时刻的观测向量	无
F_t	卡尔曼滤波模型中系统矩阵	无
H_t	卡尔曼滤波模型中观测矩阵	无
w_t	卡尔曼滤波模型中系统噪声	无
v_t	卡尔曼滤波模型中量测噪声	无
Q	卡尔曼滤波模型中系统噪声方差	无
R	卡尔曼滤波模型中量测噪声方差	无
$\phi, \theta, \psi, t_x, t_y, t_z$	基于三维刚体变换的卡尔曼滤波模型中三个方向的欧拉角和三个方向的位移	无
x_w, y_w, z_w,x_c, y_c, z_c	三维刚体中原坐标与新坐标	无
$\boldsymbol{T_{3\times1}} = (t_x, t_y, t_z)^T$	三维位移向量	无
$R_{3\times3}$	三维旋转矩阵	无
$x(k-1\|k-1)$	$k-1$ 时刻的最优解	无
$x(k\|k-1)$	利用 $k-1$ 状态的预测结果	无
$P(k\|k-1)$	$x(k\|k-1)$ 对应的协方差	无
$K(k)$	k 时刻的卡尔曼增益	无
α	低通滤波的系数	无

五　模型的建立与求解

5.1　数据预处理

图 1: 数据预处理流程图

将 924 帧图像的 json 文件转换成 DataFrame 形式后，按照题目中的示意图（图 2）的编号对应为“鼻子”，“脖子”，“右肩” 等人体特征点，并结合每列的三个数据得到 924 行，54 列的表格，每行表示某一帧中 18 个关节点的特征点捕获情况，每列依次为各关节点的横坐标、纵坐标、置信度。

5.1.1　使用插值算法处理缺失值与异常值

观察运动示例可知，在视频的某些帧捕获的关节点的坐标出现了异常，异常产生的原因主要分为以下三种情况：

1. 某些帧由于遮挡或识别异常等原因并未捕获到该关节点的 2d 坐标导致数据缺失，即相应的横坐标、纵坐标、置信度均为 0。

2. 某些序列片段身体的左右关节点识别反了导致 2d 坐标明显异常。

3. 某些帧将其他物体误认为是人体的关节点导致数据异常，如视频左侧的机械臂。

所幸题目中给出了关节点 2d 坐标的置信度水平，本文据此对缺失点与异常点按如下步骤进行初步的数据处理。

1. 将置信度水平排序，筛选出第 10 帧-第 914 帧数据中置信度位于后 5% 的数据点，进行三次样条插值替换。其中，前后第 10 帧及后 10 帧数据点不进行替换是为了避免插值时存在的边界异常问题。此步处理可以有效地解决上述原因 1 和缓解原因 2 导致的异常。

2. 经过第一步的处理仍不能完全处理好左右臂特征点交换的情况，因此再次针对 630-660 帧的数据进行线性插值替换双臂异常的数据点。画出 600-720 帧的特征点线图进行插值处理前后对比，如图 1 所示。蓝色代表原异常数据，橙色代表平滑后数据。

5.1.2 使用移动平均法平滑数据

客观世界中人体关节的运动应该是连续的，使用视频对连续运动的关节点进行时间序列离散采样具有一定失真度，而且题目已给的特征点检测数据具有明显的抖动性，无疑加剧了数据的不平滑性。因此本文考虑在对 2d 坐标数据进行平滑处理，使用移动平均法，设定滑动窗口长度为 15 帧，对前后 15 帧的横纵坐标数据取平均值作为该帧各关节点的 2d 坐标，达到数据平滑的效果。

5.1.3 检测预处理效果

鉴于眼耳特征点对该题中 3 维运动重建影响不大，因此选取除眼耳外的 14 个特征点的横纵坐标为 y 轴，以帧数为 x 轴作图检测数据预处理后的效果。如下图所示，蓝色代表原数据，橙色代表后数据，可见经过预处理后数据平滑度更高，使得人体关节连续运动产生的时间序列连续性更好。

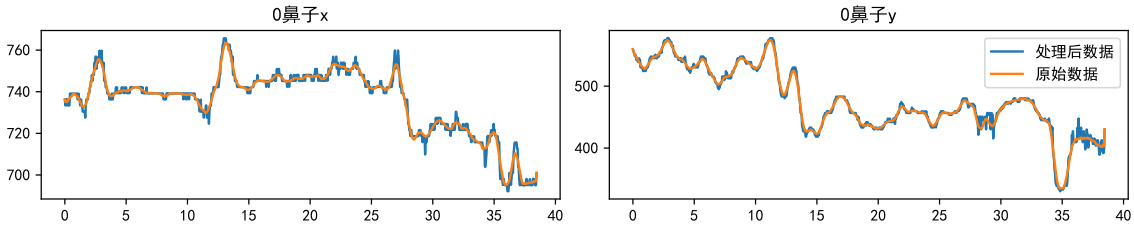

图 2: 鼻子 2D 坐标数据预处理结果

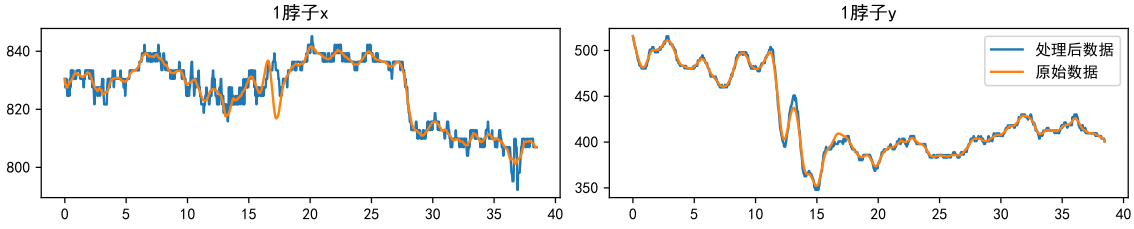

图 3: 脖子 2D 坐标数据预处理结果

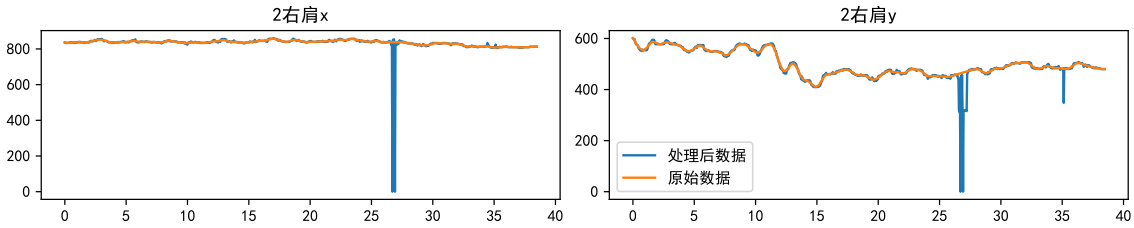

图 4: 右肩 2D 坐标数据预处理结果

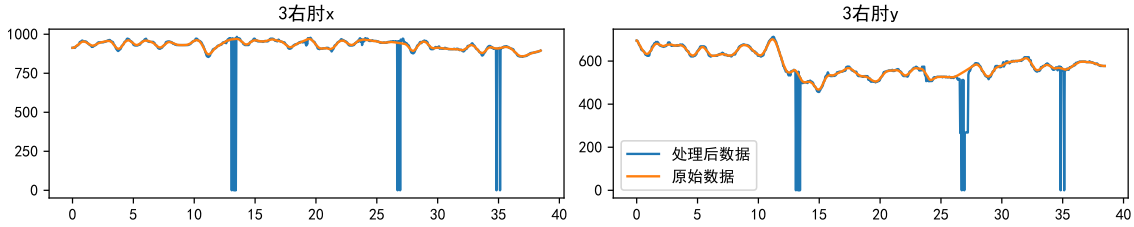

图 5: 右肘 2D 坐标数据预处理结果

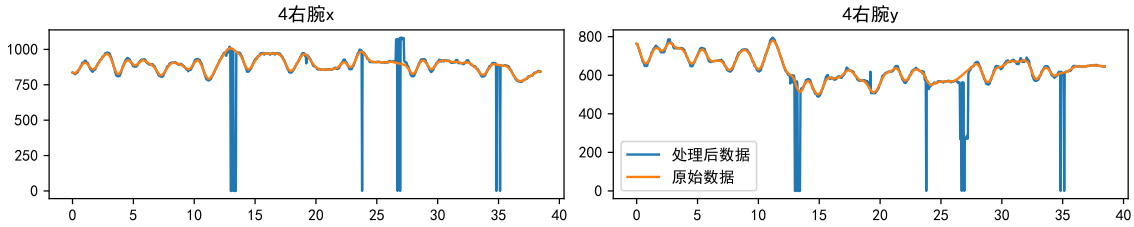

图 6: 右腕 2D 坐标数据预处理结果

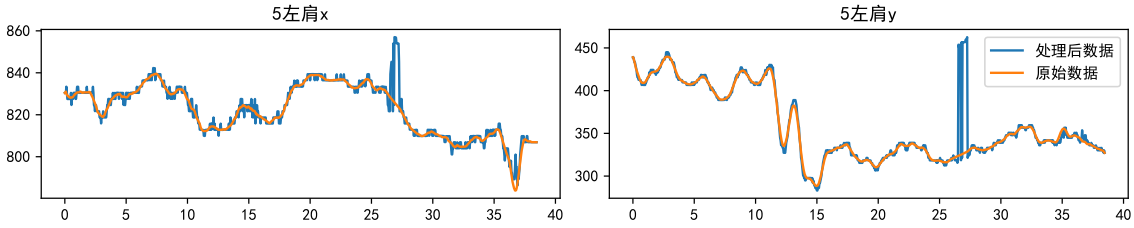

图 7: 左肩 2D 坐标数据预处理结果

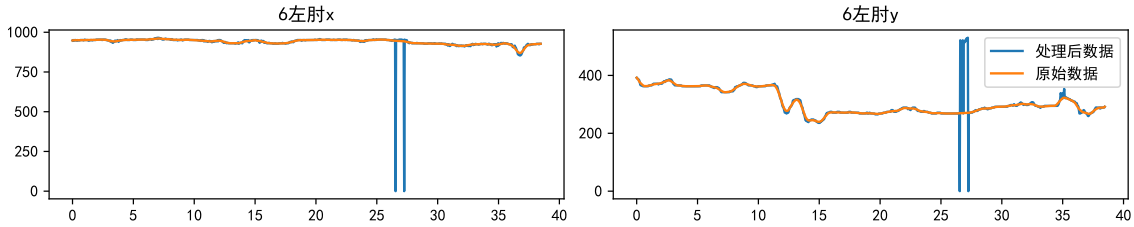

图 8: 左肘 2D 坐标数据预处理结果

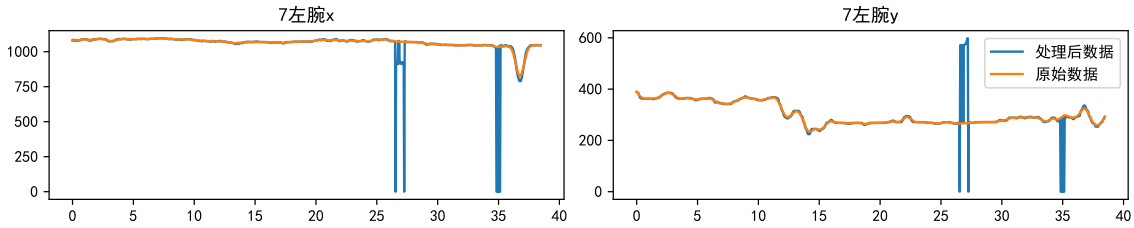

图 9: 左腕 2D 坐标数据预处理结果

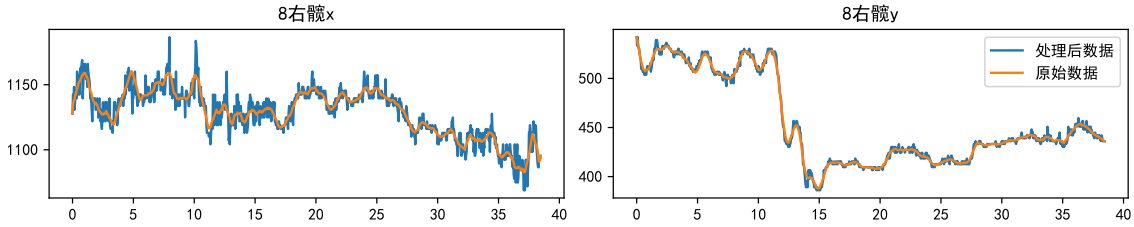

图 10: 右髋 2D 坐标数据预处理结果

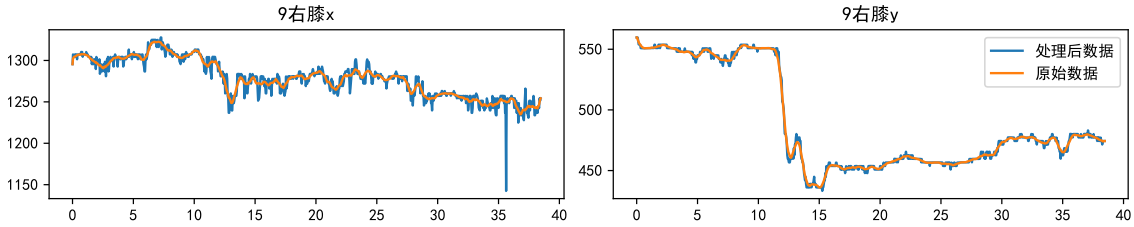

图 11: 右膝 2D 坐标数据预处理结果

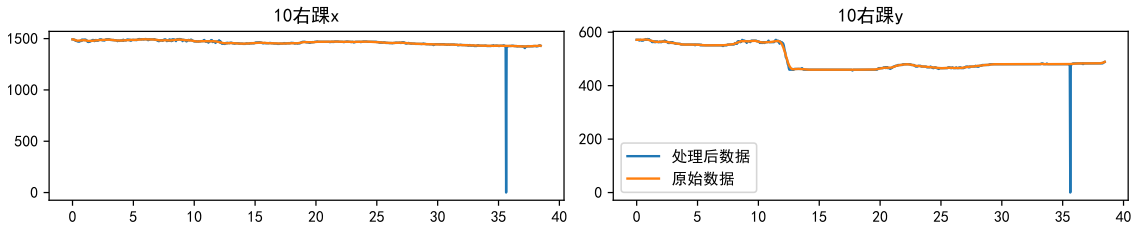

图 12: 右踝 2D 坐标数据预处理结果

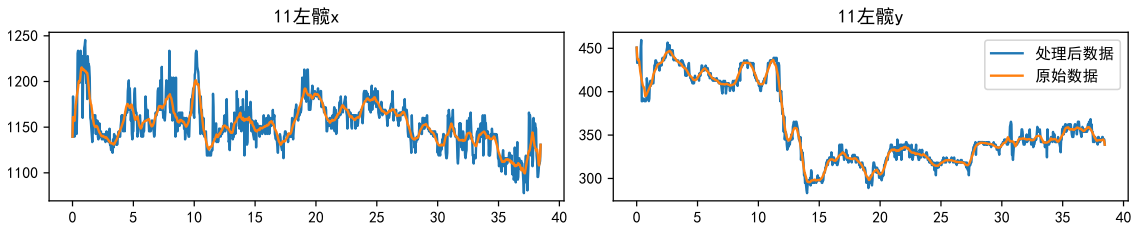

图 13: 左髋 2D 坐标数据预处理结果

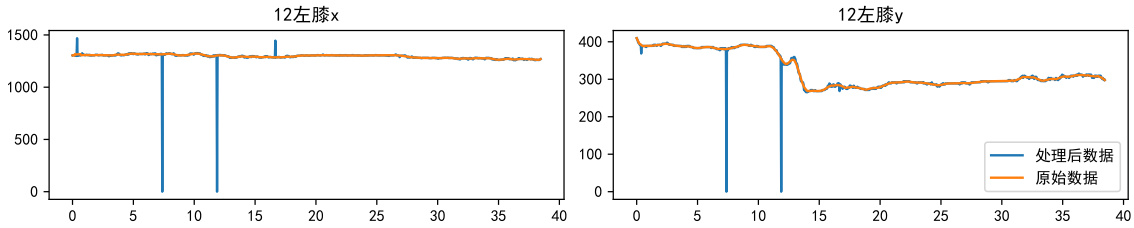

图 14: 左膝 2D 坐标数据预处理结果

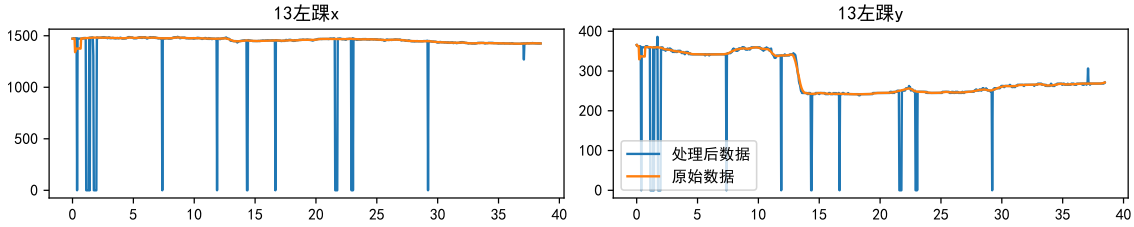

图 15: 左踝 2D 坐标数据预处理结果

5.2 模型的建立

根据问题分析一节，本题需要利用已知 924 帧中的关节点图像坐标以及对应连接关系，恢复同单位的 A 点三维人体运动序列。首先考虑到人体眼耳与所需重建的右腕 A 的运功轨迹关系不大，因此本文对题干所给的含 18 个关节点的骨骼模型简化为了只含 14 个关节点且以根节点为 1（脖子）的**三维树状骨骼模型**。通过人工选取合适的图像帧，以确定各段刚体的长度比例作为已知参数；接着，基于预处理后的人体关节点的二维坐标序列输入，我们充分利用人体各段链杆长度保持不变的约束和各关节间的骨骼连接关系，构建**基于比例正交投影的三维重建模型**，在单帧图像中进行参数估计，估计出每帧图像中链杆在比例正交投影下的比例因子。然后利用三维空间中的勾股定理计算出每个关节点的相对深度值，利用比例系数还原出骨骼模型根节点的绝对深度值进行单帧三维重建，分三步还原每一帧图像的三维像素坐标；最后，我们利用卡尔曼滤波及低通滤波算法对三维坐标序列进行优化，增强了运动序列的连续性，更加符合真实世界的运动场景。整个三维重建过程以预处理后的横纵坐标数据为输入，以还原后的三维运动轨迹为输出，流程图如图 2。

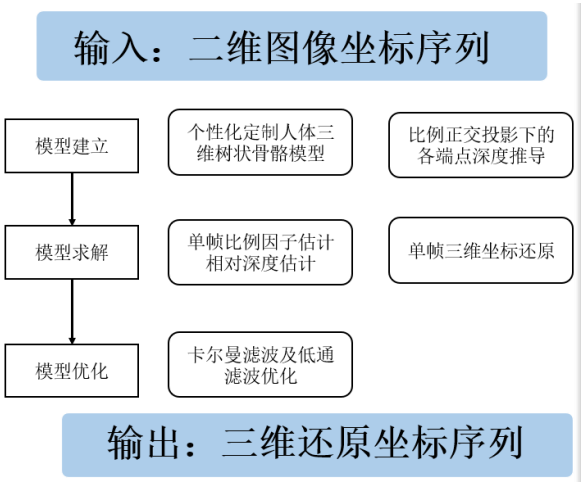

图 16: 三维重建流程图

5.2.1 人体三维树状骨骼模型

题干给出的骨骼模型由序号为 0-17 这 18 个关节点构成，将图像中的人体骨骼框架近似为球棍模型，将人体近似成一系列刚体的集合，每段刚体两端的人体关键点两两相对位置固定，具有固定的欧氏距离。由于需要还原 A 点的运动轨迹，我们更关注脸部以下的链杆运动姿态，于是我们选定运动较稳定的关节点 1（脖子）作为树形结构的根节点，舍弃关节点 14-17，个性化定制基于本视频的人体三维树状骨骼模型，如图 3 所示。其中关节点 1 作为人体的全局位置，在整体运动中确定人整体的三维轨迹，其余编号为 2-13 的 12 个关节点都有唯一一条到根关节点 1 的通路。

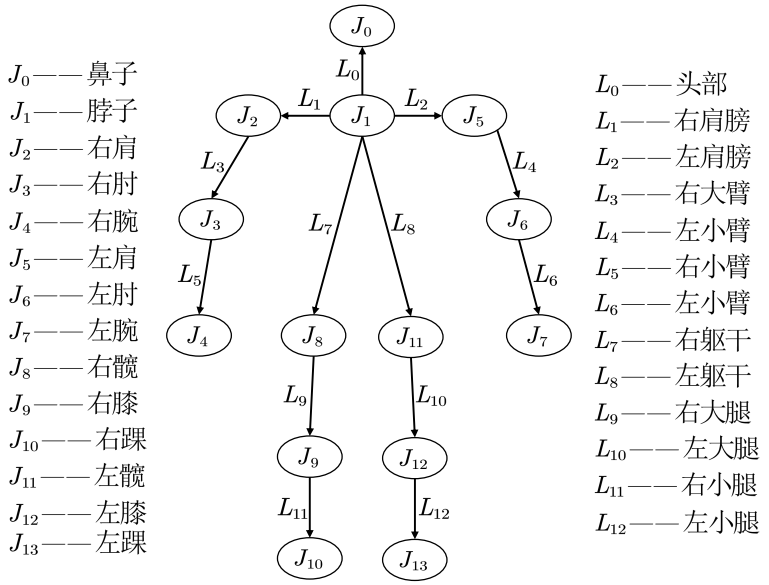

图 17: 人体三维树状骨骼模型

5.2.2 比例正交投影模型

由投影成像机理可知，图像上每一点的位置与空间物体表面的相应点几何位置关系由摄像机成像的几何模型决定。最简单的针孔模型是物体通过透视投影模型成像到图像平面，其参数标定主要有两种方法，其一是用标定板等方法人工标定摄像机的内部参数；其二是采用摄像机自标定的方式，需要多角度拍摄的图像和 5 对以上的静止特征点。因此，不借助其他先验知识，对于本题的静止摆放的单目摄像机拍摄的单目视频，我们无法在透视投影下准确重建由未定标的摄像机获取的单目视频中的人体运动。

在本题中，观察到关节之间的相对深度差 $|dZ|$ 和摄像机与人体之间的距离的比值很小，且人体根节点脖子的深度值浮动范围较小，因此本文采用比例正交投影模型来近似实际成像的透视投影模型。比例正交投影可以看成是一种“弱透视”投影，在几何上等同于正交投影加上模拟三维物体成像结果随距离收缩的效果的一个比例缩放。在比例正交投影下，人体模型上关节点的三维坐标 (X,Y,Z) 与在图像投影中的横纵坐标 (u,v) 间可以用以下方程表示，其中 s 为投影缩放的比例因子。

$$\begin{pmatrix} u \\ v \end{pmatrix} = s \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \tag{1}$$

从上式可以看出，在不考虑深度 Z 的情况下，比例正交投影的作用效果实际上只是空间坐标的比例变化，其比例因子就是上式中的参数 s。本文的重建方法就是先估计比例因子，再依据比例因子计算出人体关节的绝对深度和相对深度，实现对人体运动的 3D 重建。下面依次分析单段链杆、三段链杆、人体骨骼链杆的比例正交投影模型。

1. 单杆模型下的比例正交投影先从最简单的单段链杆模型开始分析，其比例正交投影示意图如下：

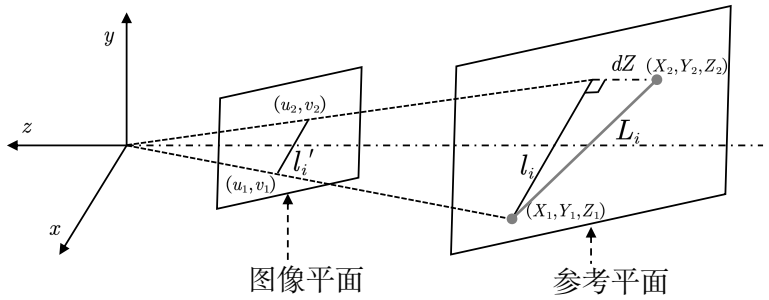

图 18: 单杆比例正交投影模型示意图

图 18中，设链杆 i 的长度为 L_i，两个端点 (X_1,Y_1,Z_1) 和 (X_2,Y_2,Z_2) 在摄像机拍摄的图像平面的投影分别为 (u_1,v_1) 及 (u_2,v_2)，链杆在参考平面的投影长度为 l_i，在图像平面的投影长度为 l_i'，投影的比例因子设为 s，链杆相对于投影平面的深度

设为 dZ。根据 (1) 式可以得到：

$$\begin{cases} X = u/s \\ Y = v/s \end{cases} \tag{2}$$

因此有：

$$\begin{cases} (u_1 - u_2) = s\,(X_1 - X_2) \\ (v_1 - v_2) = s\,(Y_1 - Y_2) \end{cases} \tag{3}$$

结合三维空间中的勾股定理，链杆的实际长度 L_i 与其两端点的坐标有如下关系：

$$L^2 = (X_1 - X_2)^2 + (Y_1 - Y_2)^2 + (Z_1 - Z_2)^2 \tag{4}$$

结合 (3)(4) 两式可以得到如下结果：

$$\begin{aligned} dZ &= \pm\sqrt{L^2 - \left((X_1 - X_2)^2 + (Y_1 - Y_2)^2\right)} \\ &= \pm\sqrt{L^2 - \left((u_1 - u_2)^2 + (v_1 - v_2)^2\right)/s^2} \end{aligned} \tag{5}$$

相对深度 dZ 的计算根据 (5) 式，我们计算得到链杆相对于投影平面的深度 dZ 的绝对值，但正负号存在歧义性。比例正交投影的歧义性如下图所示：

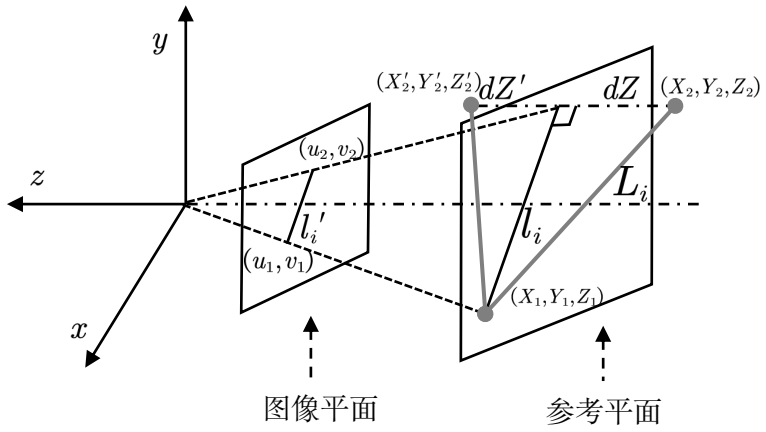

图 19: 比例正交投影产生的歧义性

图 19中，对于过点 X_1, Y_1, Z_1 的参考平面，其两侧的 X_2, Y_2, Z_2 点与 X_2', Y_2', Z_2' 在图像平面的投影点均为 (u_2, V_2)，因此产生了歧义性。从观察者的角度看，可以从视频中关节点离观察者距离的远近来决定相对深度 dZ 的符号。本文基于此基础上另外采用了人体生物力学模型进行了估计。由于人体的肘关节、膝关节的弯曲方向受到生物力学的限制，当还原出的 3D 模型不符合人体生物力学时可进行调整。

比例因子 s 的约束由于 dZ 代表端点间的相对深度，其值为实数，固有 (1) 式恒成立：

$$dZ^2 = L^2 - (X_1 - X_2)^2 + (Y_1 - Y_2)^2 \geq 0 \tag{6}$$

结合 (2) 式可以得出比例因子 s 应满足如下条件：

$$s \geq \sqrt{(u_1 - u_2)^2 + (v_1 - v_2)^2}/L \tag{7}$$

2. 三段杆模型下的比例正交投影将上述推导推广至通过四个关节点连接的三段杆模型中，如图 20所示，

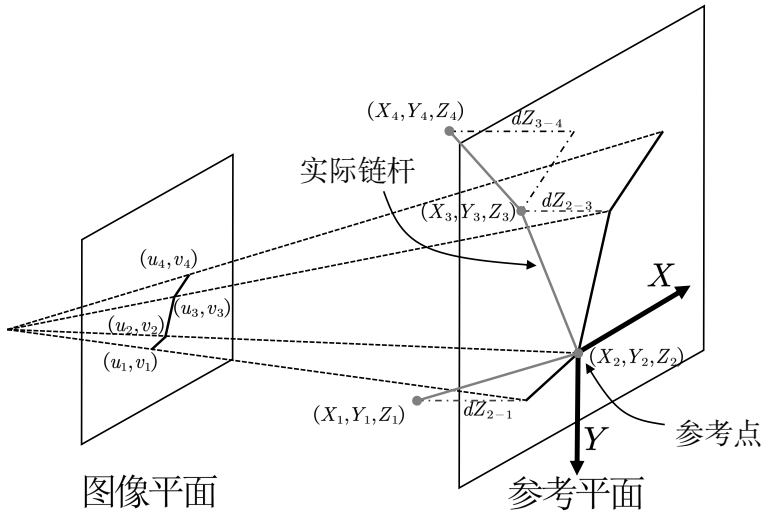

图 20: 三段杆比例正交投影模型示意图

设四个关节点 P_1, P_2, P_3, P_4 的坐标分别为 $(X_1, Y_1, Z_1), (X_2, Y_2, Z_2), (X_3, Y_3, Z_3), (X_4, Y_4, Z_4)$, 成像点分别为 $(u_1, v_1), (u_2, v_2), (u_3, v_3), (u_4, v_4)$,每段杆的比例因子分别设为 s_1, s_2, s_3。以 P_2 为参考点构造平行于图像的投影平面。同理，有对每段杆的比例因子 s_i, 需满足 (8) 式的约束条件：

$$\begin{cases} s_1 \geq \sqrt{\left((u_1 - u_2)^2 + (v_1 - v_2)^2\right)}/L_1 \\ s_2 \geq \sqrt{\left((u_2 - u_3)^2 + (v_2 - v_3)^2\right)}/L_2 \\ s_3 \geq \sqrt{\left((u_3 - u_4)^2 + (v_3 - v_4)^2\right)}/L_3 \end{cases} \tag{8}$$

考虑到同一幅图像帧上的比例因子应当相同，设为 s_0，则 s_0 满足

$$\begin{cases} s_0 \geq s_1 \\ s_0 \geq s_2 \\ s_0 \geq s_3 \end{cases} \tag{9}$$

假设参考点 P_2 的相对深度为 0，根据运动示例视频与人体生物力学约束标注好三个相对深度的正负号后，根据 (5) 式，我们可以用 s_0 表示出各端点的相对深度 dZ_1,dZ_2,dZ_3, 从而再根据 (2) 式得到各关节点的 X 坐标和 Y 坐标关于 s_0 的一元函数。换而言之，只要估计出了 s_0 即可还原出人体各关节的三维坐标。

3. 人体骨骼模型下的比例正交投影将上述三段链杆模型进一步推广到图 17所示的人体模型中，由常识可知，人体的手臂、腿脚、头部中的关节均有较大的自由活动空间，而人体的脖子活动空间有限，因此本文选取关节点 1(脖子) 即人体树状骨骼模型中的根节点作为参考点，其所在的与图像平面平行的平面为参考平面，构建比例正交投影模型。对每一个骨骼用 (7) 式计算出比例因子应满足的约束条件。对于整个人体模型而言，考虑到人体里摄像机较远，因此可近似认为每一帧中一次投影成像只对应一个比例因子 s，s 应满足由 13 个不等式约束的不等式组，若已知 s 即可由 (2) 式与 (7) 式求出各个关节点在此帧相对于参考点的相对深度 dZ。

5.2.3 单帧参数估计

在上一节的模型得，对 924 帧的每一帧提供的关节点的二维坐标，均可求解出 13 个骨骼对应的比例因子所满足的约束条件。对第 k 帧的骨骼 L_i，设比例因子为 $s_{ki}, (k = 1, 2, \cdots, 924, i = 0, 1, \cdots, 12)$。若选定了同一帧中投影的比例因子 s_k，便可以还原第 k 帧中每段骨骼中两个关节点的相对深度。考虑到视频中的人体绝大部分骨骼与图像平面平行，本文取 $s_k = \max\{s_{ki}\}\ i = 0, 1, \cdots 12$。

5.2.4 单帧三维重建

根据单帧参数估计中每一帧图像的比例因子 s，接下来我们需要求解每一帧中人体树状骨骼框架里 14 个端点的三维坐标。综合上述分析，将比例正交投影模型运用在人体骨骼框架上还原各端点的运动轨迹，分为三个步骤：

(a) 计算每帧参考点（即根节点 1）的 Z 轴深度，获得第 k 帧的参考点深度 Z_{k1}；

设相机焦距为 f，由比例正交投影模型有 $s_k = f/Z_{k2}$，由于未知相机焦距，鉴于比例性质不失一般性地假设 $f = 1$，则有 $Z_{k1} = 1/s_k$。

(b) 在根节点 1 的参考系下，根据估计出的 s 与 (5) 式还原每个端点相对于根节点的相对深度 dZ_{ij} 并与参考点深度叠加得到每个结点绝对深度由上述模型，以关节点 1-2-3-4 组成的链杆模型为例，可以通过如下的三个式子分别算出三段链杆的相对深度 dL：

$$\begin{cases} dZ_{12} = \pm\sqrt{L_1^2 - \left((X_1 - X_2)^2 + (Y_1 - Y_2)^2\right)} \\ dZ_{23} = \pm\sqrt{L_3^2 - \left((X_2 - X_3)^2 + (Y_2 - Y_3)^2\right)} \\ dZ_{34} = \pm\sqrt{L_5^2 - \left((X_3 - X_4)^2 + (Y_3 - Y_4)^2\right)} \end{cases} \tag{10}$$

上述正负号由人工标注结合人体生物力学约束确定。

根据根节点到每个端点的唯一通路，从 (a) 中估计得到的参考点的绝对深度出发，沿着树状人体骨骼逐段加入父节点到该关节点的相对深度，即

$$Z_j = Z_i + direction \times dZ_{ij} \tag{11}$$

上式中 $direction$ 为标定的方向，dZ_{ij} 为关节点 i 与 j 的相对深度，Z_i, Z_j 表示得到的关节点 i 与 j 的 Z 方向坐标，将第 k 帧中关节点 i 的 Z 方向坐标记为 Z_{ki}

(c) 在地面参考系下，还原每个关节点的 X,Y 方向坐标由式 (2), 可利用第 k 帧中第 i 个关节点的横纵坐标 (u_{ki}, v_{ki}) 求出对应在参考平面上的 X,Y 方向上的坐标 (X_{ki}, Y_{ki}):

$$\begin{cases} X_{ki} = u_{ki}/s_k \\ Y_{ki} = v_{ki}/s_k \end{cases} \tag{12}$$

由此得到地面参考系下的三维像素坐标 (X_{ki}, Y_{ki}, Z_{ki})。

5.2.5 卡尔曼滤波模型

卡尔曼滤波作为一种面向线性系统的最优估计算法，最初应用于控制领域，其根据系统输入及输出观测数据，对系统状态进行最优估算。由于观测数据中含有系统噪声等影响，所以最优估算过程也可看做滤波过程。卡尔曼滤波是一种以最小二乘法为基础来估计系统状态的方法，即将前一时刻的预报误差反馈到当前时刻的预报方程中，及时修正预报方程系数，以提高下一时刻的预报精度。由于人体关节的运动是连续的，而对视频进行特征点检测得到的特征点坐标波动幅度较大，轨迹不够平滑，因此可以采用卡尔曼滤算法将前一段时间的人体姿态反馈到当前时刻去估计当前帧的人体姿态，从而达到使运动序列平整化的目的。在卡尔曼滤波修正过程中，需确定系统的状态方程和量测方程

$$\begin{cases} x_t = F_t x_{t-1} + B u_{t-1} + w_t \\ y_t = H_t x_t + v_t \end{cases} \tag{13}$$

上式中，x_t 表示 t 时刻的状态向量，y_t 表示 t 时刻的观测向量，F_t 表示系统矩阵，H_t 表示观测矩阵，w_t 和 v_t 表示系统噪声和量测噪声, 服从均值为 0，方差分别为 Q 和 R 的高斯分布，即 $w_t \sim N(0, Q), v_t \sim N(0, R)$。对于基于卡尔曼滤波算法对人体姿态的修正，本文提出了两种方法：

1. **基于三维刚体变换的卡尔曼滤波**

   可以将人体主干脖子-左髋-右髋认为是一个刚体，基于卡尔曼滤波器对主干 3D 关节点序列进行平滑处理。刚体在三维空间中具有三维旋转平移不变性，刚体变换有对应的六个参数:$\phi, \theta, \psi, t_x, t_y, t_z$，分别对应三个方向的欧拉角和三个方向的位移

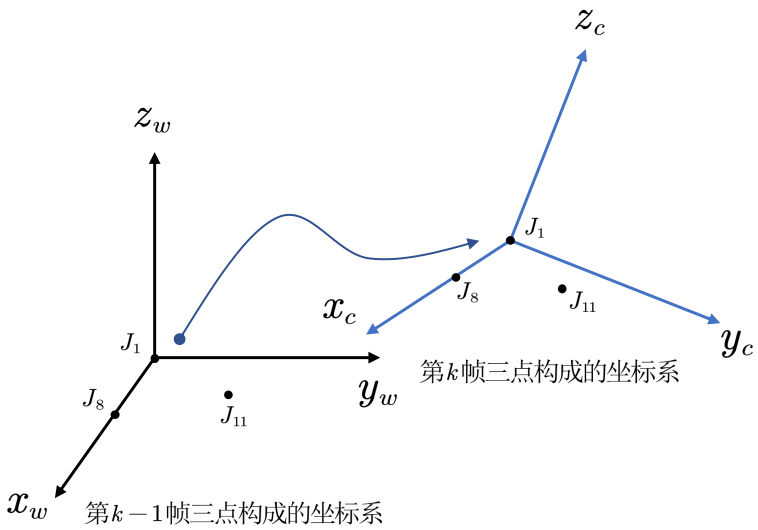

图 21: 三维刚体变换示意图

上图表示了从 x_w, y_w, z_w 坐标系经旋转与平移变换后得到新坐标系 x_c, y_c, z_c 的示意图，根据向量的点乘与叉乘，空间中三个点可以计算出一个右手系，一个三维关节点 A 的坐标 $\boldsymbol{a} = (a_x, a_y, a_z)$ 刚体变换后得到点 A' 坐标 $\boldsymbol{a'} = (a'_x, a'_y, a'_z)$。将三维向量写成齐次坐标的形式，三维刚体变换可表示为:

$$\begin{bmatrix} \boldsymbol{a'} \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}_{3\times3} & \boldsymbol{T}_{3\times1} \\ \boldsymbol{0}_{1\times3} & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{a} \\ 1 \end{bmatrix} \tag{14}$$

其中 $\boldsymbol{T_{3\times1}} = (t_x, t_y, t_z)^T$ 为三维位移向量，$R_{3\times3}$ 是由 ϕ, θ, ψ 构成的三维旋转矩阵。

$$\boldsymbol{R_{3\times3}} = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \tag{15}$$

上式中 r_i 是关于 ϕ, θ, ψ 的函数，ϕ, θ, ψ 可由坐标计算出来。三维刚体变换过程中的每一个关节点 t 时刻的状态向量 x_t 即为 $\boldsymbol{a}$，第 k 帧目标状态方程为

$$x_k = \boldsymbol{R_k}(\boldsymbol{I} + \beta(k-1))x_{k-1} + T_k + \Delta(k-1) \tag{16}$$

其中,R_k 为旋转矩阵，T_k 为位移向量，$\beta(k-1), \Delta(k-1)$ 分别为旋转和平移噪声。

2. **直接对每一维坐标进行卡尔曼滤波**

以对 x 坐标进行卡尔曼滤波为例，u_t 表示现在状态的控制量，由于该过程中没有对 x 坐标的控制量，所以 u_t 恒为 0，令状态转移矩阵 F 为单位阵，这样处理与方法一相比可以简化计算量，计算更快速且编程更为容易，因此本文采取方法二建模，此时状态方程可以简化为

$$x_t = x_{t-1} + w_t \tag{17}$$

卡尔曼滤波的具体计算步骤如下:

(a) 根据上一状态值对预测现在的状态:

$$x(k|k-1) = x(k-1|k-1) \tag{18}$$

上式中，$x(k-1|k-1)$ 是上一时刻的最优解，$x(k|k-1)$ 是利用上一状态预测的结果。

(b) 更新协方差

$$P(k|k-1) = FP(k-1|k-1)F^T + Q \tag{19}$$

此处 $P(k|k-1)$ 是 $x(k|k-1)$ 对应的协方差，$P(k-1|k-1)$ 是 $x(k-1|k-1)$ 对应的协方差，Q 为过程噪声。由于这里是一维数据，F 为 1，所以可简化为:

$$P(k|k-1) = P(k-1|k-1) + Q \tag{20}$$

(c) 计算当前的卡尔曼增益

$$K(k) = \frac{P(k|k-1)H^T}{HP(k|k-1)H^T + R} = \frac{P(k-1|k-1) + Q}{P(k-1|k-1) + Q + R} \tag{21}$$

上式中 $K(k)$ 为 k 时刻的卡尔曼增益，推导见参考文献 [11]，R 为量测噪声的方差，由于是一维数据，因此 $H^T = 1$。

(d) 修正估计值，得出最优解

$$x(k|k) = x(k|k-1) + K(k)(x_k - x(k|k-1)) \tag{22}$$

上式中 $x(k|k)$ 为当前修正值，得到当前的最优解，x_k 为第 k 帧还原出的人体 3 维坐标中的 x 坐标，$K(k)$ 为第 (3) 步计算出的卡尔曼增益。

(e) 计算最优解所对应的协方差

$$P(k|k) = (1 - K(k)H)P(k|k-1) = R \times \frac{P(k-1|k-1) + Q}{P(k-1|k-1) + R + Q} \tag{23}$$

5.2.6 低通滤波模型

采用一阶低通滤波再对经过卡尔曼滤波处理后的结果进行优化，以 x 方向的坐标为例，公式为

$$x_k = \alpha \times x_{k-1} + (1 - \alpha) \times x_{k-2} \tag{24}$$

上式中 α 为低通滤波的系数，x_k 为第 k 帧中该关节点的 x 坐标。本文选取待优化帧的前 6 帧联合起来为当前帧进行低通滤波。

5.3 模型的求解

5.3.1 个性化定制树状骨骼模型

我们先利用 2d 长度探索数据分布情况。针对构建的人体三维树状骨骼模型，我们分别算出每一帧中人体双侧的肩、大臂、小臂、躯干、大腿、小腿的 2d 长度。接着，我们分左右两侧作出箱型图，如图 9。其中，肩的 2d 长度分布最集中（集中在 70-80 单位左右）且左右均衡；大腿、小腿的 2d 长度分布较分散，左右略有差距但差距较小（大腿左右相差约 8 个单位，小腿约 20 单位）。大臂、小臂右侧数据较左侧更分散，这是因为视频中只有右侧人体大幅度运动。而小臂、躯干左右长度差距较大，小臂相差超过 50 单位，躯干超过 30 单位。然后，我们画出长度随帧数变化图，如图 10。由图可见，手臂、腿部数据波动较大，左右臂 2d 长度差距也较大。

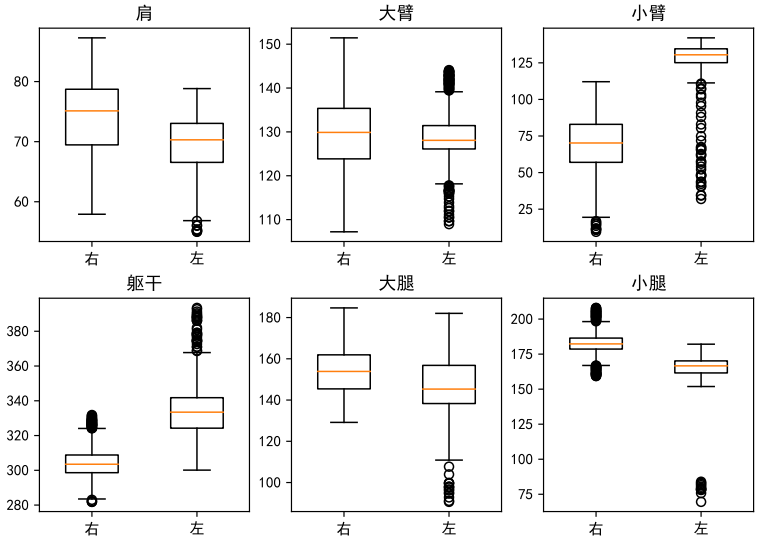

图 22: 各链杆 2d 长度分布箱型图

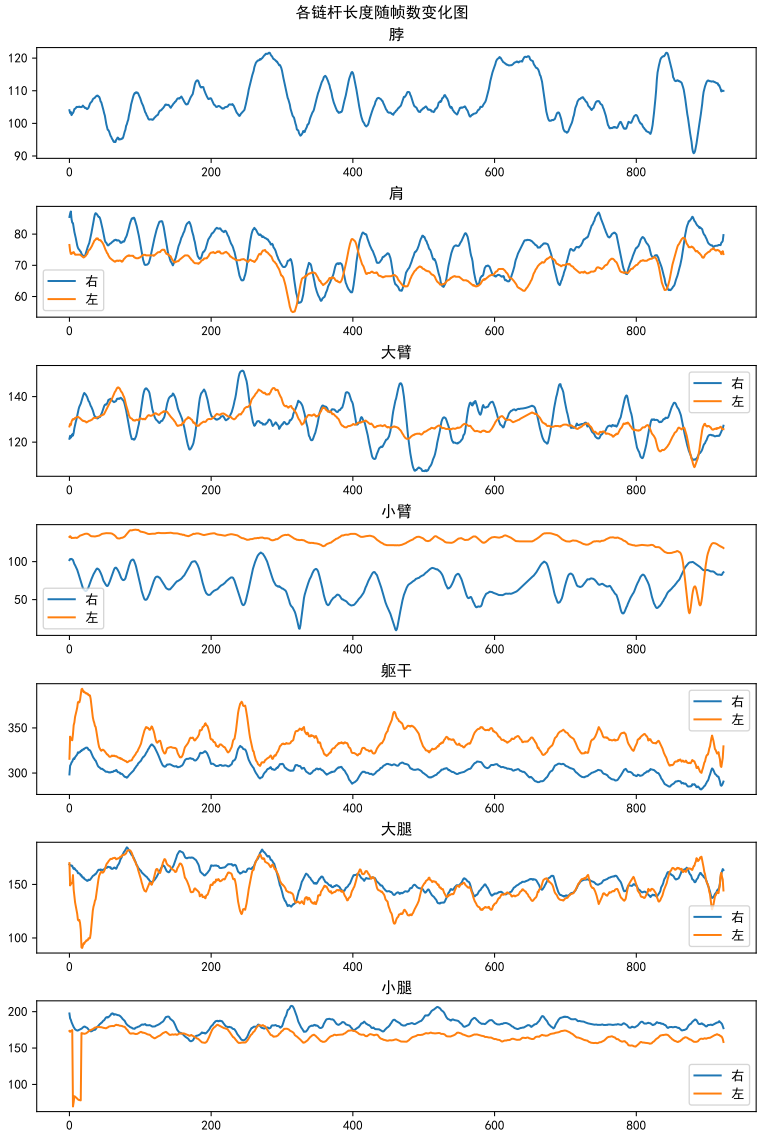

图 23: 各链杆 2d 长度随帧数变化图

由于人体骨骼长度的确定性，同人的同个视频中骨骼框架中每一段长度的相对比例应保持不变，再加上未知人体高度等其余信息，我们只需标定每一段长度的相对比例。此步骤即为根据视频及二维数据，以二维输入的像素单位作为基准，确定骨骼框架中各段链杆长度相对于身高的比例，作为后续求解的已知参数。于是，要个性化定制本题人体骨骼的相对长度，我们对每一段链杆，人工选取视频中相应链杆平行于摄影平面的若干图像帧，求平均值得到每段的相对比例长度，单位与 json 文件中二维坐标的像素单位相同。同时，考虑到后续还原需要以视频内人体身高作为基准，我们选取人体平面近似平行于图像平面的 667 帧，用视频软件测量出各链杆与铅直线的夹角，再根据生物统计学的人体脸部及颈部构造，还原人体身高（以像素单位表示），各长度汇总如表 2 所示。

表 2: 人体骨骼框架相对比例长度

颈部	**右半肩**	**左半肩**	**右大臂**	**左大臂**	**右小臂**	**左小臂**
107.0471497	73.68940186	69.38381136	129.6480363	128.8358221	128.4208722	128.4208722
右躯干	**左躯干**	**右大腿**	**左大腿**	**右小腿**	**左小腿**	**身高**
304.0159378	334.4008086	154.2654931	147.0957418	182.652963	166.8124024	815.4586077

结合数据预处理一节的数据分布探索，考虑到人体结构具有对称性，但表 2 中小腿及躯干的 3d 长度左右差距较大，考虑到轨迹还原的精确性，我们不采取左右统一长度的操作，分别利用标定好的 3d 相对长度比例作为后续求解步骤中的常量，而由于视频中右小臂一直处于弯曲状态，平均值无法很好的反映它的相对长度，因此我们将左小臂的相对长度赋给右小臂。

5.3.2 三维重建

1. 获取参考点（即根节点 1）的 Z 轴运动轨迹：由于相对性，不妨假设摄像机焦距 f 为 1，那么每一帧的参考点的深度 $Z_{k1} = 1/s_k$，同时利用 (2) 式求得参考点的横纵坐标 X_{k1}，Y_{k1}，因此得到根节点的三维坐标。

2. 根据之前建立的人体三维树状骨骼模型图，从根节点出发，利用之前求解得到的各链节相对深度值，依次求解每个节点的深度，最后还原出所有节点的三维坐标。

3. 根据所有节点的三维坐标，建立三维人体骨架模型图，以第 600 帧，第 700 帧为例，我们在三维坐标系中画出重建后的人体骨架模型，如图 24 , 图 25所示。

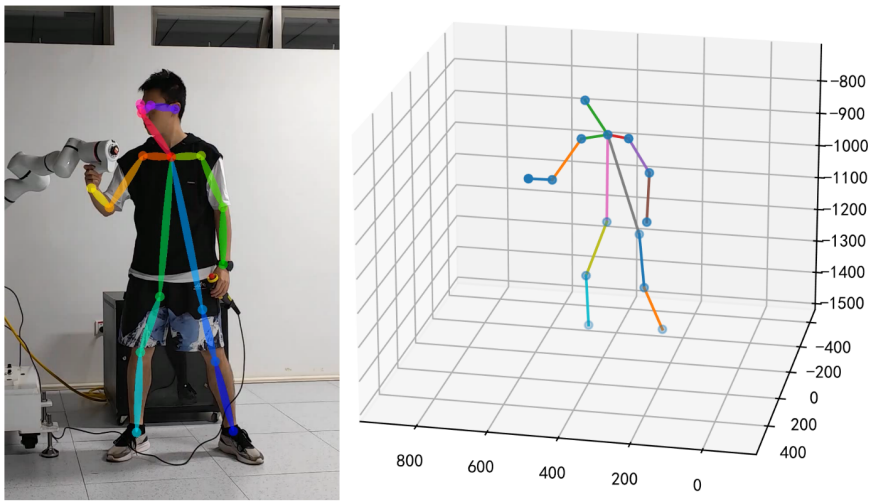

图 24: 第 600 帧重建得到的 3D 人体姿态图

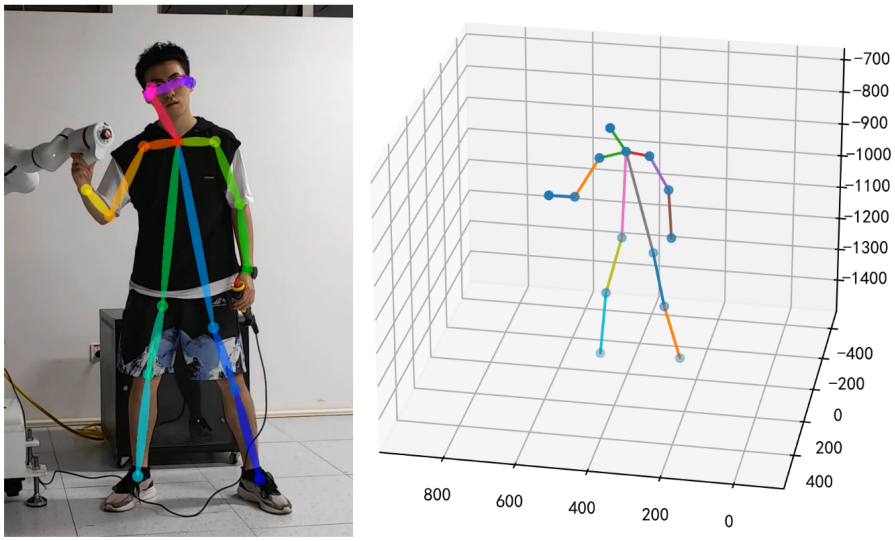

图 25: 第 700 帧重建得到的 3D 人体姿态图

5.3.3 A 点运动轨迹还原

1. 利用 (2) 式，将右腕节点（即 A 点）的三维坐标同时乘以当前帧比例因子 s_k，还原得到 A 点以摄像机为原点，以像素为单位的三维坐标。

2. 建立以 A 点为原点的参考系，此坐标系以原题图像数据的横纵坐标系为 X、Y 轴，以摄像机镜头朝向为 Z 轴正向。

3. 将 A 点以摄像机为原点，以像素为单位的三维坐标变换到以 A 点为原点，像素为单位的三维坐标，即所有 A 点的三维坐标减去初始 A 点的三维坐标，因此得到了 A 点相对于初始状态的运动轨迹。

5.4 模型的优化

在单帧三维重建后，利用卡尔曼滤波及低通滤波对三维坐标进行优化，使其更符合人体运动的连续性。将原始数据、仅使用卡尔曼滤波处理的结果、使用卡尔曼滤波与低通滤波综合处理的结果对比图如下：

从图26中可以明显看出，卡尔曼滤波算法与低通滤波算法消去了序列中的某些尖锐点，使得 3D 重建结果更加平滑，更符合人体关节运动的连续性。

六 模型的评价、改进与推广

6.1 模型的优点

1. 本文模型兼具个性化与一定的普适性。

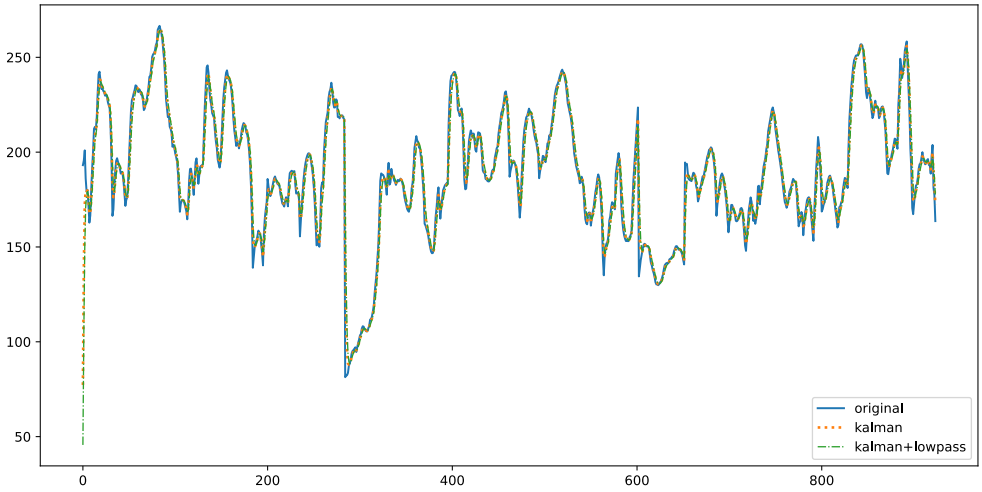

图 26: 卡尔曼滤波与低通滤波对 A 点 Z 坐标处理对比图

本文通过个性化定制人体骨骼树状模型，获取贴合视频的人体骨骼框架。同时，在模型推广一节也给出了普适性的人体框架图。

本模型不需要任何专门的辅助设备，不需要摄像机定标，对视频长短也没有限制。

2. 充分利用刚体长度不变的特性进行建模，符合题目要求。

3. 针对题干所给的特征点坐标和最后还原出的 3d 坐标分别进行滑动平均与卡尔曼滤波，使得模型所得结果更加平滑，符合真实世界中人体运动的连续性。

6.2 模型的缺点

1. 当人体关节点在整个视频中长时间与参考平面夹角较大时需人工指定关节点长度

2. 只能求出 3d 重建后坐标的相对比例，而不是绝对长度

3. 使用正交比例投影近似相机的透视投影在人像距离摄像机较近时误差较大

4. 面对 dZ 的二义性视频的部分帧下只能人工指定

6.3 模型的改进

6.3.1 比例因子的其他约束

我们可以构造 s_j 的约束使得满足“投影平面与图像平行”及“投影平面过根节点 1”这两个限制条件。由于在标定长度时，选取视频中相应链杆平行于摄影平面的若干图像帧，所以我们可以利用过根节点 1 的某段链杆（设其编号为 k），求解对于全局 924 帧

的全部比例因子 $s_{kj}, j = 1\ 2\ \cdots\ \cdots 924$。那么“投影平面过根节点 1”这一条件可以通过 $s_j \geq s_{kj}, j = 1\ 2\ \cdots\ \cdots 924$ 来限制。同时，为满足 $s_j \geq \max\{s_{ij}\}, i = 0, 1, \cdots\ \cdots 13$，我们取 $s_j = \max\{\max\{s_{ij}\}, \max\{s_{kj}\}\}$ 作为第 j 帧的参数估计。由图的 2d 长度分布箱图，左右肩的 2d 长度（关节点 1-4 段及 1-5 段）随帧数变化最稳定且左右均衡，我们取 k=1 及 2，取 $\max\{s_{1j}\}$ 和 $\max\{s_{2j}\}$ 的平均值作为 $\max\{s_{kj}\}$ 的取值进行后续估计。

6.3.2 s 的灵敏度分析

我们知道，每一幅图像的比例因子 s 代表根据投影原理对各坐标的调整系数，其取值会影响所有链杆和参考平面的夹角。s 取值越大，相应求解出的 dZ 也越大，所有链杆和参考平面的夹角也越大。因此，可以根据不同的 s 取值，作出对应的人体框架图，观察比例因子取值对三维轨迹还原的效果。

6.3.3 获取更多信息进行适用范围的扩展

1. 如果有多目摄像头或者两张不同视角的图片，可以计算出摄像机的参数，从而获取真实世界中的人物高度与关节长度，而不是本模型计算得到的比例

2. 如果有更多的 2d 数据集与相应已知的三维坐标可采用深度学习的方式进行训练，结合本文提出的方法进行综合优化。

3. 可以将获取的运动轨迹辅以纹理、光照等图像其余信息，生成逼真的动画。

6.4 模型的推广

6.4.1 普适性人体骨骼框架

在确定人体骨骼模型的相对比例时，对于视频序列中不存在相应链杆平行于摄影平面图像帧的情况，我们可以采用生物力学统计模型中的人体测量学参数，下表为适合中国等东亚人群的测量数据。（数据来自《人体解剖学》，郭连起，1994）

6.4.2 参考点运动轨迹还原

在已知图像比例因子，进行单帧三维重建时，如果人体骨骼模型的根节点深度有变化，需要先人工给出参考点的 Z 轴运动序列，再利用以上各链杆的深度求解模型还原在静止参考系下的相对深度。

参考文献

[1] 陈坚. 单目视频人体运动跟踪和获取技术研究 [D]. 中国科学院研究生院（软件研究所）,2005.

[2] 陈姝. 基于视频的人体运动跟踪与重构方法研究 [D]. 中南大学,2008.

[3] 视觉 SLAM 十四讲：从理论到实践高翔等著.—北京：电子工业出版社，2017.3

[4] Martinez, J., Hossain, R., Romero, J., & Little, J. (2017). A Simple Yet Effective Baseline for 3d Human Pose Estimation. 2017 IEEE International Conference on Computer Vision (ICCV), 2659-2668.

[5] Remondino, F., & Roditakis, A. (2004). Human Motion Reconstruction and Animation from Video Sequences.

[6] Taylor, C.J. (2000). Reconstruction of articulated objects from point correspondences in a single uncalibrated image. Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662), 1, 677-684 vol.1.

[7] Zhao, J., Li, L., Chee, K., & Keong (2004). A Model-based Approach for Human Motion Reconstruction from Monocular Images.

[8] 邱茂林, 马颂德, 李毅. 计算机视觉中摄像机定标综述 [J]. 自动化学报,2000(01):47-59.DOI:10.16383/j.aas.2000.01.006.

[9] 佟帅, 徐晓刚, 易成涛, 邵承永. 基于视觉的三维重建技术综述 [J]. 计算机应用研究,2011,28(07):2411-2417.

[10] 敬喜. 卡尔曼滤波器及其应用基础 [M]. 北京: 国防工业出版社，1973.

[11] Kálmán, R.E. (1960). A new approach to linear filtering and prediction problems" transaction of the asme journal of basic.

[12] https://zhuanlan.zhihu.com/p/108422696

[13] https://blog.csdn.net/Asimov_Liu/article/details/96442990

附录

附录 1：数据预处理与计算骨骼长度:

```python
 # data_processing.py
import json
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.interpolate import CubicSpline

plt.rcParams['font.family'] = 'SimHei' #设置字体
plt.rcParams['axes.unicode_minus']=False #坐标轴的负号正常显示

keypoints_2d = []
for i in range(924):
    num_str = str(i).zfill(3)
    path = "./key_points/Motion_000000000%s_keypoints.json"%num_str
    with open(path, 'r') as fp:
        keypoints_2d.append(json.load(fp)['people'][0]['pose_keypoints_2d'])

df = pd.DataFrame(keypoints_2d)
df.to_excel("keypoints.xlsx")

bodyparts =
    ["鼻子","脖子","右肩","右肘","右腕","左肩","左肘","左腕","右髋","右膝",
"右踝","左髋","左膝","左踝","右眼","左眼","右耳","左耳"]
column_easy_read = []
for bodypart in enumerate(bodyparts):
    column_easy_read.append(str(bodypart[0])+bodypart[1]+'x')
    column_easy_read.append(str(bodypart[0])+bodypart[1]+'y')
    column_easy_read.append(str(bodypart[0])+bodypart[1]+'conf')

df_column_changed = df.copy()
df_column_changed.columns = column_easy_read
df_column_changed


def dot_fill_low_conf(df, columns_id):
    df_cp = df.copy()
```

```
    low_conf =
        (df.iloc[:,columns_id[2]].sort_values(ignore_index=True))[int(924*0.05)]
    num_for_subsititute_id = np.where(df.iloc[:,columns_id[2]] <=
        low_conf)[0]
    normal_id = df.index.drop(num_for_subsititute_id)
    CS_x = CubicSpline(normal_id, df.iloc[normal_id,columns_id[0]])
    df.loc[num_for_subsititute_id, columns_id[0]] =
        CS_x(num_for_subsititute_id)
    CS_y = CubicSpline(normal_id, df.iloc[normal_id,columns_id[1]])
    df.iloc[num_for_subsititute_id, columns_id[1]] =
        CS_y(num_for_subsititute_id)
    CS_conf = CubicSpline(normal_id, df.iloc[normal_id, columns_id[2]])
    df.iloc[num_for_subsititute_id, columns_id[2]] =
        CS_conf(num_for_subsititute_id)
    df.iloc[:10,columns_id] = df_cp.iloc[:10,columns_id]
    df.iloc[-10:,columns_id] = df_cp.iloc[-10:,columns_id]
    return df

def fill_low_conf(df):
    for i in range(18):
        dot_fill_low_conf(df, [3*i,3*i+1,3*i+2])

df_fill_low_conf = df.copy()
fill_low_conf(df_fill_low_conf)
df_fill_low_conf

# 计算关节之间的2d距离
def cal_joints_lenths_2d(df, joints_id):
    joint0xy = df.loc[:,[joints_id[0]*3, joints_id[0]*3+1]]
    joint1xy = df.loc[:,[joints_id[1]*3, joints_id[1]*3+1]]
    lengths_2d = ((joint0xy.values -
        joint1xy.values)**2).sum(axis=1)**(1/2)
    return lengths_2d

# 对630到660帧之间双臂错误位置进行线性插值
def dot_fill_error(df, columns_id):
    df_cp = df.copy()
    num_for_subsititute_id = df.index[630:660]
    normal_id = df.index.drop(num_for_subsititute_id)
    df.iloc[num_for_subsititute_id, columns_id[0]] =
```

```
        np.interp(num_for_subsititute_id, normal_id, df.iloc[normal_id,
        columns_id[0]])
    df.iloc[num_for_subsititute_id, columns_id[1]] =
        np.interp(num_for_subsititute_id, normal_id, df.iloc[normal_id,
        columns_id[1]])
    df.iloc[num_for_subsititute_id, columns_id[2]] =
        np.interp(num_for_subsititute_id, normal_id, df.iloc[normal_id,
        columns_id[2]])
    return df

def fill_error(df):
    for i in range(2,8):
        dot_fill_error(df, [3*i,3*i+1,3*i+2])

df_fill_error = df_fill_low_conf.copy()
fill_error(df_fill_error)
df_fill_error

row = 6
col = 12 // row
for j in range(2,8):
    i = j - 2
    fig,axes = plt.subplots(1, col, figsize=(10,3), layout='constrained')
    axes[0].set_title(df_column_changed.columns[3*j])
    axes[0].plot(df.index[600:720],df.iloc[600:720,3*j])
    axes[0].plot(df.index[600:720],df_fill_error.iloc[600:720,3*j])
    axes[1].set_title(df_column_changed.columns[3*j+1])
    axes[1].plot(df.index[600:720],df.iloc[600:720,3*j+1])
    axes[1].plot(df.index[600:720],df_fill_error.iloc[600:720,3*j+1])
    plt.savefig('./630到660帧异常值图/%s.svg'%str(j),dpi=200)
    plt.show()

# 自定义的滑动平均函数，arr是np.array类型，window_size为奇数
def move_average(arr, window_size):
    new_arr = np.convolve(arr, np.ones((window_size,))/window_size,
        mode='same')
    for i in range(window_size//2):
        new_arr[i] = arr[:2*i+1].sum()/(2*i+1)
        new_arr[len(arr)-1-i] = arr[-2*i-1:].sum()/(2*i+1)
    return new_arr
```

```
df_move_average = df_fill_error.copy()
df_move_average = df_move_average.apply(lambda x:move_average(x,15),
    axis=0)
df_move_average.to_excel('keypoints(processed).xlsx')
df_move_average
row = 14
col = 28 // row
fig,axes = plt.subplots(row, col, figsize=(10,20), layout='constrained')
for i in range(14):
    axes[2*i//col, 2*i%col].title.set_text(df_column_changed.columns[3*i])
    axes[2*i//col, 2*i%col].plot(df.index/24,df[3*i])
    axes[2*i//col, 2*i%col].plot(df.index/24,df_move_average[3*i])
    axes[2*i//col,
        2*i%col+1].title.set_text(df_column_changed.columns[3*i+1])
    axes[2*i//col, 2*i%col+1].plot(df.index/24,df[3*i+1])
    axes[2*i//col, 2*i%col+1].plot(df.index/24,df_move_average[3*i+1])
plt.show()

joints_pair = [[[1,2],[1,5]],
               [[2,3],[5,6]],
               [[3,4],[6,7]],
               [[1,8],[1,11]],
               [[8,9],[11,12]],
               [[9,10],[12,13]]]
bone_title = ['肩','大臂','小臂','躯干','大腿','小腿']

fig, axes = plt.subplots(7,1, figsize=(8,12), layout='constrained')
axes[0].set_title('脖')
axes[0].plot(cal_joints_lenths_2d(df_move_average, [0,1]))
for i in range(6):
    axes[i+1].set_title(bone_title[i])
    axes[i+1].plot(cal_joints_lenths_2d(df_move_average,
        joints_pair[i][0]), label='右')
    axes[i+1].plot(cal_joints_lenths_2d(df_move_average,
        joints_pair[i][1]), label='左')
    axes[i+1].legend()

plt.suptitle('各链杆2d长度随帧数变化图')
plt.savefig('各链杆2d长度随帧数变化图.svg', dpi=200)
plt.show()
```

```
139
140 joints_pair = [[[1,2],[1,5]],
141                [[2,3],[5,6]],
142                [[3,4],[6,7]],
143                [[1,8],[1,11]],
144                [[8,9],[11,12]],
145                [[9,10],[12,13]]]
146 bone_title = ['肩','大臂','小臂','躯干','大腿','小腿']
147 fig, axes = plt.subplots(2,3, figsize=(7,5), layout='constrained')
148 for i in range(6):
149     axes[i//3, i%3].set_title(bone_title[i])
150     axes[i//3, i%3].boxplot([cal_joints_lenths_2d(df_move_average,
        joints_pair[i][0]),cal_joints_lenths_2d(df_move_average,
        joints_pair[i][1])], labels=['右','左'], widths=0.5)
151 plt.show()
152
153 len_ls = []
154 len_ls.append(cal_joints_lenths_2d(df_move_average, [0,1]).mean())
155 for i in range(6):
156     len_ls.append(cal_joints_lenths_2d(df_move_average,
        joints_pair[i][0]).mean())
157     len_ls.append(cal_joints_lenths_2d(df_move_average,
        joints_pair[i][1]).mean())
158 chain_rod_columns =
    ['颈部','右半肩','左半肩','右大臂','左大臂','右小臂','左小臂','右躯干',
'左躯干','右大腿','左大腿','右小腿','左小腿']
160 chain_rod_length = pd.Series(len_ls, index=chain_rod_columns)
161 chain_rod_length['右小臂'] = chain_rod_length['左小臂']
162 chain_rod_length.to_excel('各链杆相对长度.xlsx', header=False)
```

附录 2: 三维姿态还原:

```
1 # motion_reconstruction.py
2 import numpy as np
3 import pandas as pd
4 import matplotlib.pyplot as plt
5
6 plt.rcParams['font.family'] = 'SimHei' # 设置中文字体
7 plt.rcParams['axes.unicode_minus']=False # 坐标轴的负号正常显示
8 plt.rcParams['figure.constrained_layout.use'] = True # 自动约束布局
```

```
9
10 df = pd.read_excel('./keypoints(processed).xlsx', index_col=0)
11 df_chain_rod_length = pd.read_excel('./各链杆相对长度.xlsx', header=None)
12 chain_rod_length = df_chain_rod_length.iloc[:,1].values
13
14 # 计算关节之间的2d距离
15 def cal_joints_lenths_2d(df, joints_id):
16     joint0xy = df.loc[:,[joints_id[0]*3, joints_id[0]*3+1]]
17     joint1xy = df.loc[:,[joints_id[1]*3, joints_id[1]*3+1]]
18     lengths_2d = ((joint0xy.values -
           joint1xy.values)**2).sum(axis=1)**(1/2)
19     return lengths_2d
20 joints_pair = [[0,1],
21                 [1,2],[1,5],
22                 [2,3],[5,6],
23                 [3,4],[6,7],
24                 [1,8],[1,11],
25                 [8,9],[11,12],
26                 [9,10],[12,13],]
27 ls = []
28 for i in range(len(joints_pair)):
29     ls.append(cal_joints_lenths_2d(df, joints_pair[i]))
30 joints_length_arr = np.array(ls).T
31
32 s = (joints_length_arr/ chain_rod_length).max(axis=1)
33
34 dZ_unsigned = pd.DataFrame(np.around((chain_rod_length**2 -
       ((joints_length_arr**2).T / (s**2)).T),10)**(1/2))
35 dZ_sign = pd.read_excel('dZ方向.xlsx')
36 dZ = pd.DataFrame(dZ_unsigned.values * dZ_sign.values)
37
38 Z_add_path = {'0':(0,),
39         '2':(1,),
40         '3':(1,3),
41         '4':(1,3,5),
42         '5':(2,),
43         '6':(2,4),
44         '7':(2,4,6),
45         '8':(7,),
46         '9':(7,9),
```

```
        '10':(7,9,11),
        '11':(8,),
        '12':(8,10),
        '13':(8,10,12),
        }
Z0 = 1 / s
Z = np.zeros((924,14))
Z[:,1] = Z0
for key in Z_add_path:
    Z[:,int(key)] = dZ.iloc[:,list(Z_add_path[key])].sum(axis=1)

X = ((df.iloc[:,[3*i for i in range(14)]].T / s).T).values
Y = ((df.iloc[:,[3*i+1 for i in range(14)]].T / s).T).values

X = (X.T * s).T
Y = (Y.T * s).T
Z = (Z.T * s).T


def line_plot(i,X,Y,Z, joints):
    plt.plot([X[i,joints[0]], X[i,joints[1]]], [Y[i,joints[0]],
        Y[i,joints[1]]], [Z[i,joints[0]], Z[i,joints[1]]])
def skeleton(i,X,Y,Z):
    ax = plt.subplot(projection='3d')
    line_plot(i,X,Y,Z, [4,3])
    line_plot(i,X,Y,Z, [3,2])
    line_plot(i,X,Y,Z, [2,1])
    line_plot(i,X,Y,Z, [1,5])
    line_plot(i,X,Y,Z, [5,6])
    line_plot(i,X,Y,Z, [6,7])
    line_plot(i,X,Y,Z, [1,8])
    line_plot(i,X,Y,Z, [1,11])
    line_plot(i,X,Y,Z, [8,9])
    line_plot(i,X,Y,Z, [9,10])
    line_plot(i,X,Y,Z, [11,12])
    line_plot(i,X,Y,Z, [12,13])
    line_plot(i,X,Y,Z, [0,1])
    ax.scatter(X[i],Y[i],Z[i])
    ax.set_aspect('equal')
    return ax
```

```
skeleton(0, Y, Z, -X)
plt.show()
```

附录 3: 卡尔曼滤波与低通滤波:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


x = pd.read_excel("X.xlsx",index_col=0)
y = pd.read_excel("Y.xlsx",index_col=0)
z = pd.read_excel("Z.xlsx",index_col=0)
record = []
for i in range(x.shape[0]):
    for j in range(x.shape[1]):
        record.append(x.iloc[i][j])
        record.append(y.iloc[i][j])
        record.append(z.iloc[i][j])


parent =[1,-1,1,2,3,1, 5,6,1,8,9,1,11,12]
jointnum = 14
dataOri = np.reshape(record,(-1,jointnum,3))
framenum = dataOri.shape[0]


datakalman = np.zeros_like(dataOri)
datafinal = np.zeros_like(dataOri)

# 卡尔曼滤波系数
KalmanParamQ = 0.0001
KalmanParamR = 0.00015
K = np.zeros((jointnum,3),dtype=np.float32)
P = np.zeros((jointnum,3),dtype=np.float32)
X = np.zeros((jointnum,3),dtype=np.float32)
PrevPose3D = np.zeros((6,jointnum,3),dtype=np.float32)
for idx in range(framenum):
    currdata = np.squeeze(dataOri[idx])
```

```
    smooth_kps = np.zeros((jointnum,3),dtype=np.float32)
    '''
    kalman filter
    '''
    for i in range(jointnum):
        K[i] = (P[i] + KalmanParamQ) / (P[i] + KalmanParamQ + KalmanParamR)
        P[i] = KalmanParamR * (P[i] + KalmanParamQ) / (P[i] + KalmanParamQ
            + KalmanParamR)
    for i in range(jointnum):
        smooth_kps[i] = X[i] + (currdata[i] - X[i])*K[i]
        X[i] = smooth_kps[i]

    datakalman[idx] = smooth_kps # 保存卡尔曼滤波结果

    '''
    low pass filter
    '''
    LowPassParam = 0.1
    PrevPose3D[0] = smooth_kps
    for j in range(1,6):
        PrevPose3D[j] = PrevPose3D[j] * LowPassParam + PrevPose3D[j - 1] *
            (1.0 - LowPassParam)
    datafinal[idx] = PrevPose3D[5] # record kalman+low pass result.

'''
visualization
'''
x1 = np.arange(0,framenum,1)
y1 = np.squeeze(dataOri[:,4,2])
plt.plot(x1,y1)
plt.plot(x1,datakalman[:,4,2],linewidth=2,linestyle=":")
plt.plot(x1,datafinal[:,4,2],linewidth=1,linestyle="-.")
plt.legend(["original","kalman","kalman+lowpass"])
plt.show()

# visualize the skeleton animation
fig = plt.figure()
plt.ion()
data = datafinal#dataOri
for i in range(framenum):
```

```
    fig.clf()
    kps = np.squeeze(data[i,...])
    ax = fig.add_subplot(projection = '3d')
    ax.xaxis.set_label_text(label="x")
    ax.yaxis.set_label_text(label="y")
    ax.zaxis.set_label_text(label="z")
    ax.set_xlim(np.min(data[...,0]),np.max(data[...,0]))
    ax.set_ylim(np.min(data[...,2]),np.max(data[...,2]))
    ax.set_zlim(np.min(data[...,1]),np.max(data[...,1]))
    ax.view_init(elev=26., azim=70)
    ax.scatter3D(kps[:,0],-kps[:,2],kps[:,1],'red')
    for i in range(14):
        if(parent[i]!=-1):
            ax.plot3D(kps[[i,parent[i]],0], -kps[[i,parent[i]],2],
                kps[[i,parent[i]],1], 'gray')
    plt.pause(2)
plt.ioff()
plt.show()
```