Estimating 3D Human Shapes from Measurements阅读笔记

训练模型

在PCA空间中，每个三维网格X_i都由一个向量W_i表示，通过特征分析可以得到一个线性映射关系,给定一个新的测量值就可以得到对应的网格在PCA中的权重：

$$W_{new} = BP_{new} \tag{1}$$

对于每个训练集里的网格X_i进行特征分析，可以得到一个平均模型 μ 和矩阵A。有了这两个参数就可以通过一个新的模型的权重W_{new}求出对应的模型X_{new}：

$$X_{new} = AW_{new} + \mu \tag{2}$$

由公式（1）（2）给定测量值P_{new}就可以得到对应的人体模型：

$$X_{new} = ABP_{new} + \mu \tag{3}$$

模型微调

算法概述

已知通过学习方法获取的一个初始模型$X_{new}^{init} \Rightarrow$ 将W看做关于X_{new}^{init}的函数：$W_{new} = A^{+}(X_{new}^{init} - \mu)$，通过优化$X_{new}^{init}$生成更优的$W_{new}$⇒将$W_{new}$带入回归模型得到新的初始模型$X_{new}^{pca} = AW_{new} + \mu$⇒对$X_{new}^{pca}$进行相同的尺寸优化，但是要加上保形能量项$E = (1-\lambda)E_m + \lambda E_s$,得到最终结果$X_{new}$

算法细节

测量的数据分为三类，欧式距离值，测地距离值和围长值，对于每个待求的网格X_i，在网格上对应的求出的数据尽量和真实值保持一致，这就是一个最小化能量函数的问题

$$E_e = \sum_{d \in \mathcal{D}} \left((\mathbf{p_i} - \mathbf{p_j})^2 - (l_t(d))^2 \right)^2$$

$$E_g = \sum_{e \in \mathcal{P}} \left((\mathbf{p}_k - \mathbf{p}_l)^2 - (l_t(e))^2 \right)^2$$

$$E_c = \sum_{e \in \mathcal{C}} \left((\mathbf{q}_i - \mathbf{q_j})^2 - (l_t(e))^2 \right)^2 = \sum_{e \in \mathcal{C}} \left((\alpha(\mathbf{p}_n - \mathbf{p_m}) + \mathbf{p}_m - \beta(\mathbf{p_o} - \mathbf{p}\right.$$

其中p是顶点坐标向量，q是切平面与网格上三角面片的边的交点坐标向量。$l_t(d)$是实际测量线段的长度，$l_t(e)$是实际测量的围长，欧式距离就是两点坐标的直线距离直接计算就可以，而测地距离和围长是多个顶点距离之和，每一段的逼近长度需要单独计算出来。假定变形前后长度比例不变，可以通过下式算出逼近的长度：

$$l_t(d) = \frac{l_t(P)}{l_g(P)} l_g(e)$$

$$l_t(e) = \frac{l_t(C)}{l_g(C)} l_g(e)$$

在模型预测部分我们知道，只要给定一个全新的降维后的主成分W_{new}，就可以带入线性回归模型$X_{new} = AW_{new} + \mu$，得到一个全新的模型。那么接下来目标就是，如何获得一个更准确的W_{new}。我们知道$W_{new} = A^{+}(X_{new}^{init} - \mu)$,所以可以从如何获取一个比较好的$X_{new}^{init}$着手。

- Minimization with respect to W_{new}
  X_{new}^{init}可以通过学习的方法算出一个初始的模型，通过对三类尺寸的优化获得更精确的模型。优化方法采用拟牛顿法，需要对方程求导：$\nabla_{W_{new}} E = A^{+} \nabla_{p_i} E$

$$\nabla_{\mathrm{pi}} E_e = \sum_{d \in D(p_i)} 4\left(\left(\mathbf{p_i} - \mathbf{p_j}\right)^2 - \left(l_t(d)\right)^2\right)\left(\mathbf{p_i} - \mathbf{p_j}\right)$$

$$\nabla_{\mathrm{pk}} E_g = \sum_{e \in P(p_i)} 4\left(\left(\mathbf{p_k} - \mathbf{p_l}\right)^2 - \left(l_t(e)\right)^2\right)\left(\mathbf{p_k} - \mathbf{p_l}\right)$$

$$\nabla_{\mathrm{pm}} E_c = \sum_{e \in \mathcal{C}} 4\left(\left(\alpha(\mathbf{p}_n - \mathbf{p_m}) + \mathbf{p}_m - \beta(\mathbf{p_o} - \mathbf{p_m}) - \mathbf{p_m}\right)^2 - \left(l_t(e)\right)^2\right)(\alpha$$

优化欧式距离时，可以直接对p_i求导，优化测地距离时，用的是最短路径，也就是路径顶点之间距离之和，所以也可以直接对p_i求导，而优化围长时，围长与网格的交点不一定是网格顶点，所以不能直接求导。但是交点仍然可以用其所在边的两个端点来线性表示它。所以通过转换后仍然可以对p_i求导。

计算出新的W_{new}后就可以通过线性回归模型得到新的模型$X_{new}^{pca} = AW_{new} + \mu$。但是这个模型仍然是数据集空间中的模型。

- Minimization with respect to p_i
  接下来就用网格优化的方法，得到一个数据集构建的空间无法描述的全新的结果。
  如果直接对$E_m = E_e + E_g + E_c$最小化能量处理，可能会导致网格不光滑。所以加入一个平滑项来保证得到模型在人体空间内。该平滑项的意义是让相邻的顶点都有相似的变形，$\Delta \mathbf{p}_i$表示变形前后的平移向量。

$$E_s = \sum_{p_i \in X_{new}} \sum_{p_j \in N(p_i)} \left(\Delta \mathbf{p}_i - \Delta \mathbf{p}_j\right)^2$$

$$\nabla_{\mathbf{p_i}} E_s = \sum_{p_j \in N(p_i)} 2\left(\Delta \mathbf{p_i} - \Delta \mathbf{p_j}\right)$$

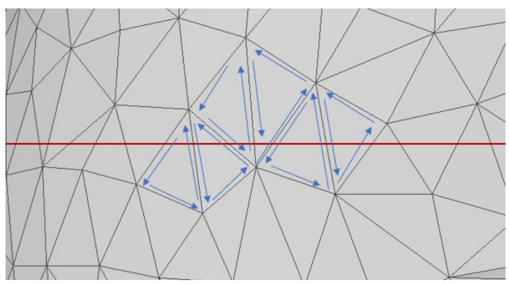
将顶点初始化为上一步中求得的X_{new}^{pca}，这一步的整体能量函数可以表示为$E=(1-\lambda)E_m+\lambda E_s$。

实现细节

求平面与模型的交点

算法概述

对每条边设置访问标记，遍历所有边，对所有边进行相交检测，找到一条相交的边$\Rightarrow$ 从该边开始，用halfedge进行遍历操作，对所在面三条边进行相交检测，如果halfedge对应边相交，跳到其对边继续进行遍历，直到对边等于起始半边$\Rightarrow$ 算法结束会得到三个点集，计算三个点集的边长和，取其最大的



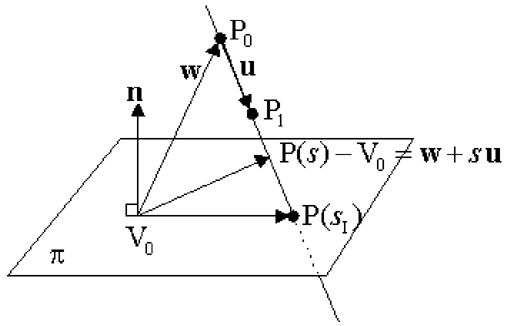
相交检测

平面与模型相交问题的本质是线段与平面相交的问题。在讨论线段与平面相交问题前，先讨论一般的情况，即直线与平面相交的问题。空间中平面π可以由顶点V和一个法向量n来描述，直线$L = P(s) = P_0 + s(P_1 - P_0) = P_0 + su$。

- 直线与平面平行
  直线与平面平行有两种情况，一种完全在平面内，一种不在平面内。暂且不考虑完全在平面内的情况。直线平行于平面数学上的表达为平面法线与直线垂直：

$$n \cdot u = 0$$

- 直线与平面相交



设交点为$P(s_i)$,$w = P_0 - V_0$,则$P(s) - V_0 = w + su$垂直于面法向n，即：

$$n \cdot (w + su) = 0$$

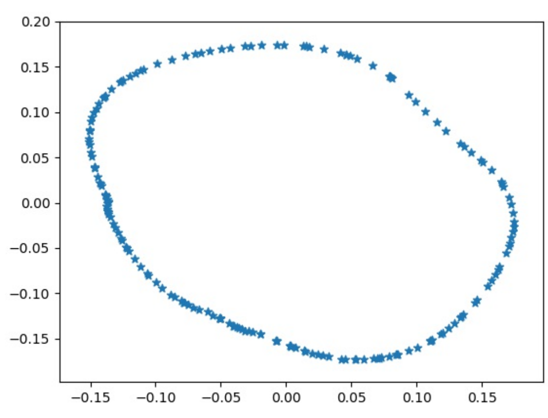
$$s_i = \frac{-n \cdot w}{n \cdot u} = \frac{n \cdot (V_0 - P_0)}{n \cdot (P_1 - P_0)}$$

从直线相交的情况，我们可以发现，只要保证s这个伸缩因子在$[0-1]$之间就能确定线段与平面相交，接下来讨论线段与平面相交的情况

- 线段与平面相交

$$0 \leq s_i \leq 1$$

点集结果

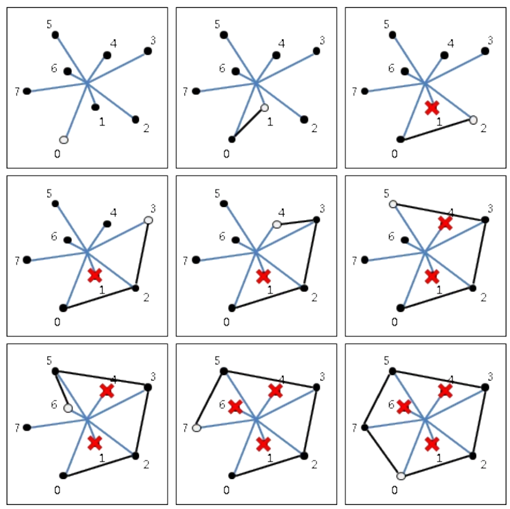


由所有交点构成凸包

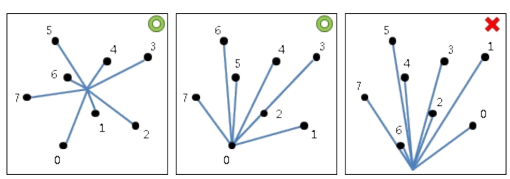
解决二维凸包问题，有很多成熟的算法，比如GrahamScan,QuickHull,Chan's Algorithm等等。GrahamScan实现最简便，但是不能很好的解决多点共线的情况。但是因为点集是近凸包的，是在模型表面收集的点，所以没有三点共线的情况，为了方便实现所以采用GrahamScan算法。

GrahamScan

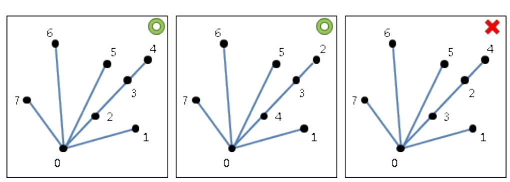
Graham's Scan尝试先将所有点依照时针顺序排好，再来做绕行-圈的动作，绕行途中顺便排除凸包内部的点，如此就不必以穷举所有点的方式来寻找最外围的点。



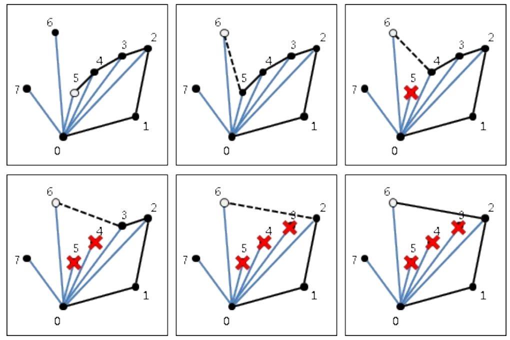
要让所有点依照顺时针顺序排好，只要将中心点设定在凸包内部或者凸包上，然后让各点依照角度排序即可。但是不能将中心点设置在凸包外部，如果把中心点设定在凸包外部，结果就不一定是顺时针顺序了。包覆时必须按照时顺时针，才能确保结果正确。



点集中的点与中心点做叉积运算，对点集的点进行排序，遇到共线的情况，按照离中心点距离排序，可以正序可以倒序，但是不可以随意排序。



每次加入新点，都要与已在凸包的两个点进行Toleft检测，如果在外侧，就加入凸包，并且弹出最上面的点，凹陷的点必定不是凸包上的顶点。如果在内测，就将其加入凸包。


6
4
3
2
5
7
1
0
6
4
3
2
5
7
1
0
6
4
3
2
5
7
1
0
6
3
2
4
5
7
1
0
6
4
3
2
5
7
1
0
6
3
2
4
5
7
1
0

问题

生成凸包算法，可能toleft检测有些问题，最后的结果不太对。