using LandmarkCollection = std::vector<Landmark<LandmarkType>>; 序号,x,y

序号从1开始，类型是string

LandmarkCollection<Eigen::Vector2f> landmarks;

68个2D关键点及序号

std::unordered\_map<std::string, std::string> landmark\_mappings

<2D landmark,3D landmark> 序号都是从1开始

此时3D landmark的初始值是基本模型上的点

image\_points 存放2D landmark

model\_points 存放3D landmark 齐次形式

vertex\_indices 存放3D landmark 序号

# 估计的投影矩阵

估算仿射矩阵，像素坐标以左下角为原点

CameraType::Orthographic



affine\_camera\_matrix： 最后一行为0,0,0,1

# 计算Shape系数

构造矩阵： , 每隔三行插一行0，使用Shape的PCA基构造

 表示主成分个数



表示所要求的系数， 是Shape的基本模型。

 ，取

令 求得线性解为



# 计算Expression系数

ModelContour:: std::vector<int> right\_contour; 3D右轮廓点序号从1开始

ModelContour:: std::vector<int> left\_contour; 3D左轮廓点序号从1开始

ContourLandmarks:: std::vector<std::string> right\_contour;2D右轮廓点序号从1开始

ContourLandmarks:: std::vector<std::string> left\_contour; 2D左轮廓点序号从1开始

左右以人本身的视角定义

EdgeTopology 包含每条边邻接的两个面的序号和两个顶点的序号

Expression系数的计算方法和Shape系数计算方法一样

此时取带表情和形状的基本模型， 使用表情的PCA基构造。

glm::vec4(0, height, width, -height);

# 迭代计算Shape系数和Expression系数

初始的2D和3D对应点为内部对应点，不包括轮廓点

## 4.1 查找未被遮挡的2D和3D轮廓对应点

遍历每一个2D轮廓点（根据偏航角过滤左右两部分轮廓点）

将3D点投影到平面，找到距离该2D点最近的3D点作为该2D点的对应3D点

这里遍历的是2D投影点

这一步完成后会将对应的2D点坐标和3D序号分别添加到image\_points、vertex\_indices用来拟合Shape系数用的。

## 4.2 查找处于边界的3D边和3D点

一个3D轮廓边的两个邻接面方向不一致则认为是边界边，两个顶点是边界顶点

## 4.3 查找处于边界的2D和3D对应点

使用被遮挡的2D轮廓点构建kdtree

将处于边界的3D点投影然后查找kdtree上的对应点

如果距离小于阈值，构成一对匹配的

<2D遮挡点，3D边界点序号>

这里遍历的是3D投影点

这一步完成后会将对应的2D点坐标和3D序号分别添加到image\_points、vertex\_indices用来拟合Shape系数用的。

## 4.4 重新估算投影矩阵

根据前三步得到的2D和3D对应点估算投影矩阵

## 4.5 拟合Shape系数和Expression系数