### 一、简介

(General Graph Optimization)[12]是一种对基于图模型的优化问题（包含图实现问题）的通用解法，对于非线性的目标函数，其算法思想为通过问题特质寻找其初始解，利用目标函数的一阶泰勒展开式将其线性化，然后对于替代后的目标函数，一般使用高斯牛顿法(Gauss– Newton)或莱文贝格－马夸特方法(Levenberg– Marquardt)[35]迭代优化。

首先介绍通用的模型。的一般模型中，定义为图中节点待求解状态（例如位置），为*i*节点和*j*节点之间的约束关系，为*i、j*两个节点的信息对约束条件的满足程度，其目标函数定义如下:

(7)

其中为关于*i、j*两个节点的信息矩阵，根据具体应用问题确定。在当前结果处一阶泰勒

展开式为

(8)

其中为函数雅克比矩阵。将一阶泰勒展开式替代原式，变换目标函数如下：

(9)

该公式中，定义, , ，均为常数项，且, , 。因此将原目标函数变换为向量的二次方程，令其导数为零，可得以下方程：

(10)

求解即可获得此次运算的迭代结果，该结果为高斯-牛顿法求解的结果，莱文贝格－马夸特方法通过在变换后方程的一阶微分中加入阻尼因子，控制收敛速度[36]，其形式如下：

(11)

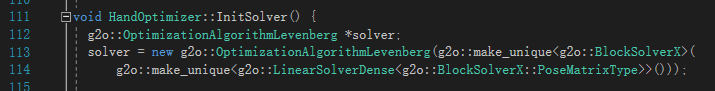
算法的求解框架适用于各种基于图模型的问题，或可以转化为图模型的问题，该方法广泛应用于机器人同步定位与地图构建问题中[37][38][39][40][41]。将图实现问题应用于算法的求解框架，*X*所表示的待求解状态即为节点的坐标，节点之间的约束即为两个节点的欧氏距离，约束信息即为距离的测量值，因此，无额外约束信息，为单位矩阵，迭代的初始解可通过三边测距法或是SDP算法求得。算法通过一阶泰勒展开式将目标函数的形式简化，因此求解结果与目标函数实际的局部最优解可能存在一定偏差，但相比于上文提及的三边测量法及求解距离方程类方法，该算法具备更高的准确性和更强的鲁棒性。

#### 二、**配置**

配置有几个要素，分别为solver、点、边、action，下面分别介绍。

1. solver

即定义使用什么优化算法来求解问题，目前使用的是Levenberg方法，简单理解为牛顿法加上迭代步长，还有梯度下降和牛顿法可选，也可定义自己的求解器。详见HandOptimizer::InitSolver()



1. 点

在中点为求解的状态，例如后处理中点保存的状态为手腕的位置、手掌旋转和手指20个旋转角度。（此外，例如SphereCamera工程中点为相机的位姿（6自由度）和特征点的位置。）

后处理中点定义详见HandOptimizer::InitVertex ()，分为两类点，g2o::VertexSE3Expmap 和 VertexAngle，其中VertexSE3Expmap是手腕的六自由度位姿，共一个，为 中预定义的点；VertexAngle是手指角度，共20个，每个点里面对应一个角度，为自己定义的点。

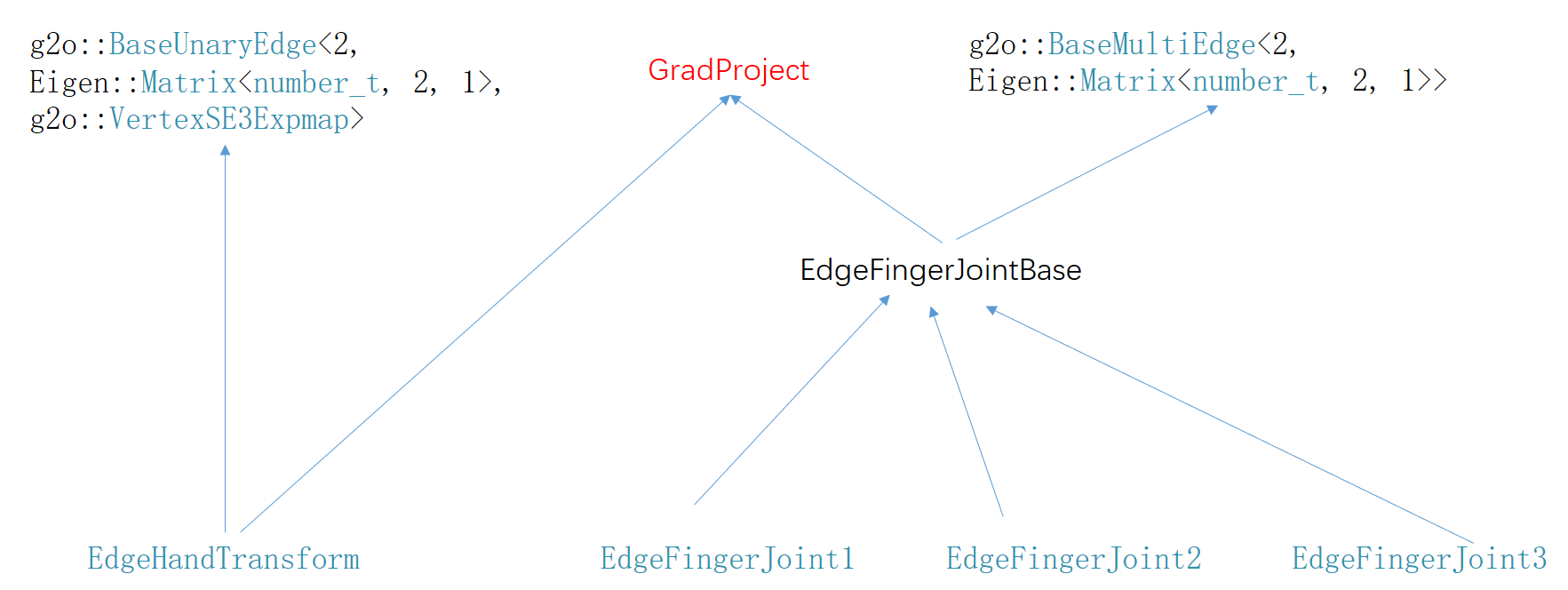
定义点时参考VertexAngle.hpp，主要是几个virtual方法的定义，关键是virtual void oplusImpl(const number\_t \*update)方法，描述了该状态如何更新 。

3、边

每条边定义为优化问题的一个误差项，可参考HandOptimizer::InitialEdges()，往优化器中加边的时候可参考HandOptimizer::InitialEdgeTransform，初始化后主要添加的信息包括id，关联的点id，所使用的参数和信息矩阵（边的权重）。

本项目中的边都是自己定义，目前使用的基类分为两种，BaseMultiEdge和BaseUnaryEdge，区别为关联的点的数量，两个类均为模板类，模板为error的维度和measurement数据类型，BaseUnaryEdge还需Vertex的类型。

以EdgeFingerJoint1为例，基类是BaseMultiEdge，measurements为投影的像素点，所以基类的类型为BaseMultiEdge<2, Eigen::Matrix<number\_t, 2, 1>>。本文中手指joint对应的边有共同的基类EdgeFingerJointBase，EdgeFingerJointBase有两个基类，一个是BaseMultiEdge，另一个基类GradProject主要定义了相机投影和SE3的求导方法。项目中边的继承关系如下：



主要的三个方法为构造函数，computeError和 linearizeOplus。构造函数可参考EdgeFingerJointBase::EdgeFingerJointBase，需要指定参数的对应关系。 computeError参考EdgeFingerJoint1::computeError，计算误差项。 linearizeOplus为求误差项对每个关联点的偏导数。由于handmodel是树状结构，以EdgeFingerJoint1为例，涉及到三个点，分别为手腕SE3,手指根部的两个角度点。链式求导时分为几个坐标系：像素坐标系中像素点M、世界坐标系中位置P、手掌坐标系中位置点Q、手指二维平面角度值θ，所以求导如下：

见class GradProject中solveGrad函数，class GradProject中dong\_SE3函数，见EdgeFingerJointBase::linearizeOplus\_joint1， EdgeFingerJointBase::linearizeOplus\_joint2， EdgeFingerJointBase::linearizeOplus\_joint3.

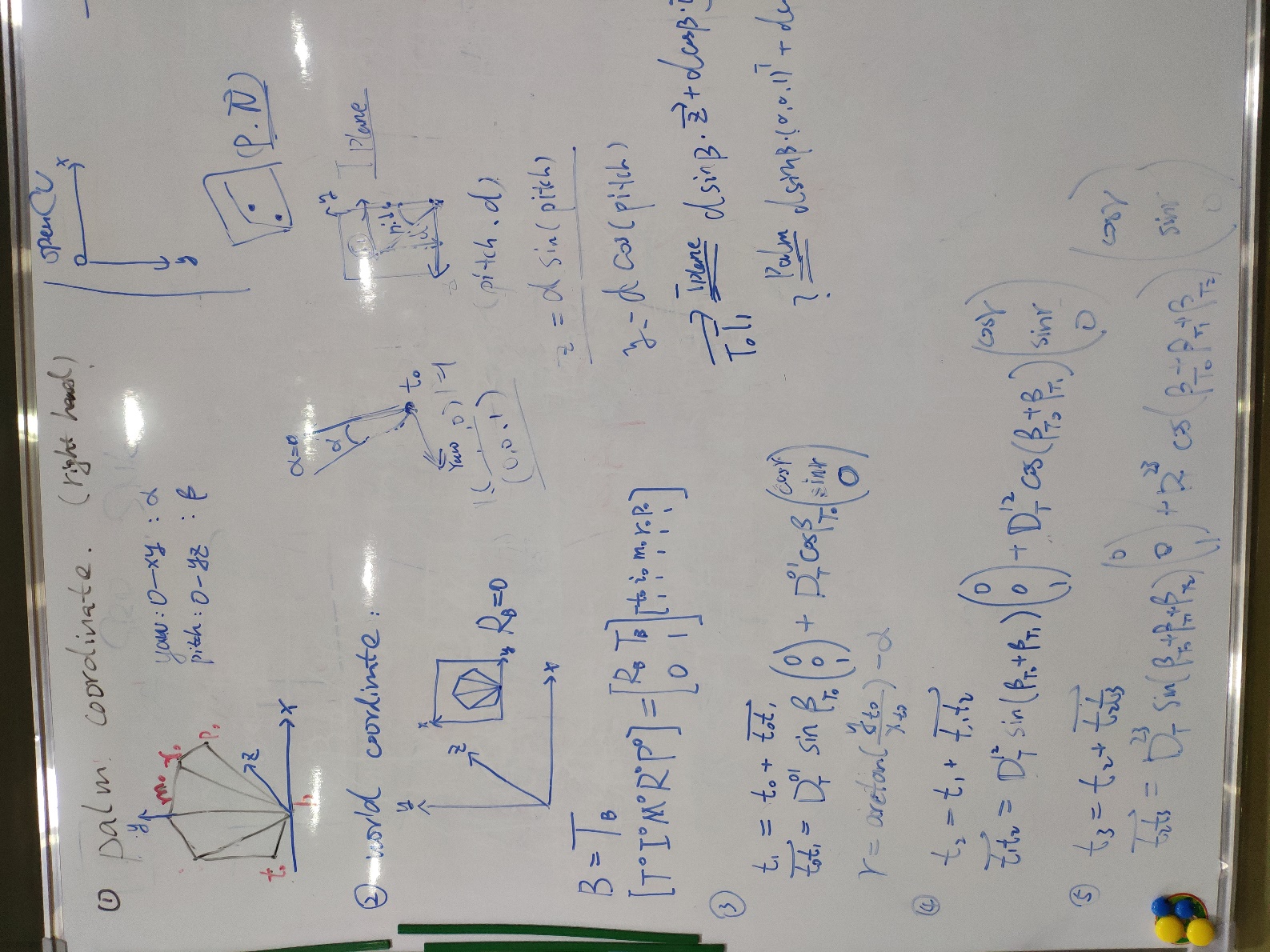
4、Action

参考 中SparseOptimizer::optimize，目前了解到的Action分为三类，preIterationAction为在每次优化开始做，ComputeErrorAction在每次计算error之前做，postIterationAction是每次迭代结束做。本项目中\*HandOptimizer::HandAction是ComputeErrorAction，功能为计算21个点的3D位置，避免重复计算；HandOptimizer::ConstraintAction 为preIterationAction，功能为在每次迭代之前将反弯曲的点修正。

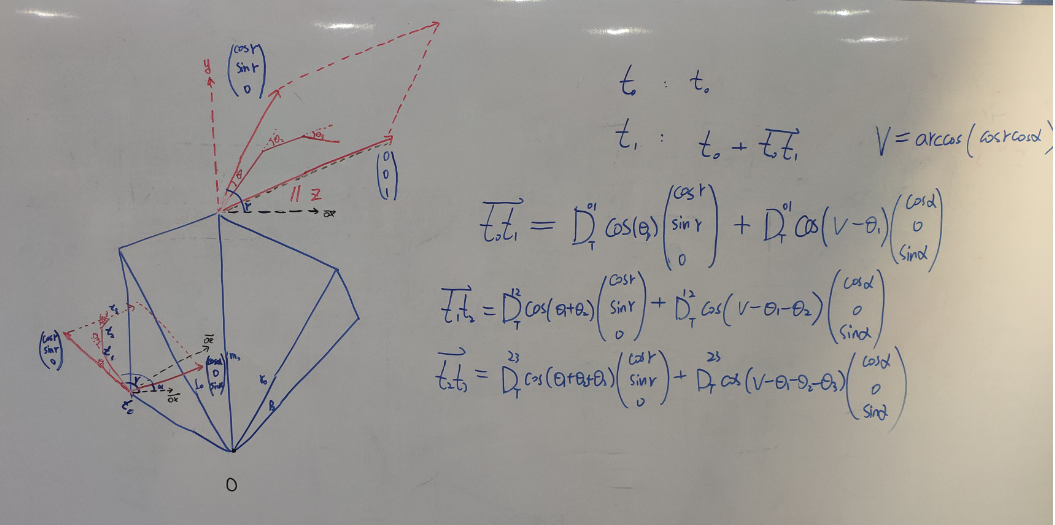


#### 三、Handmodel

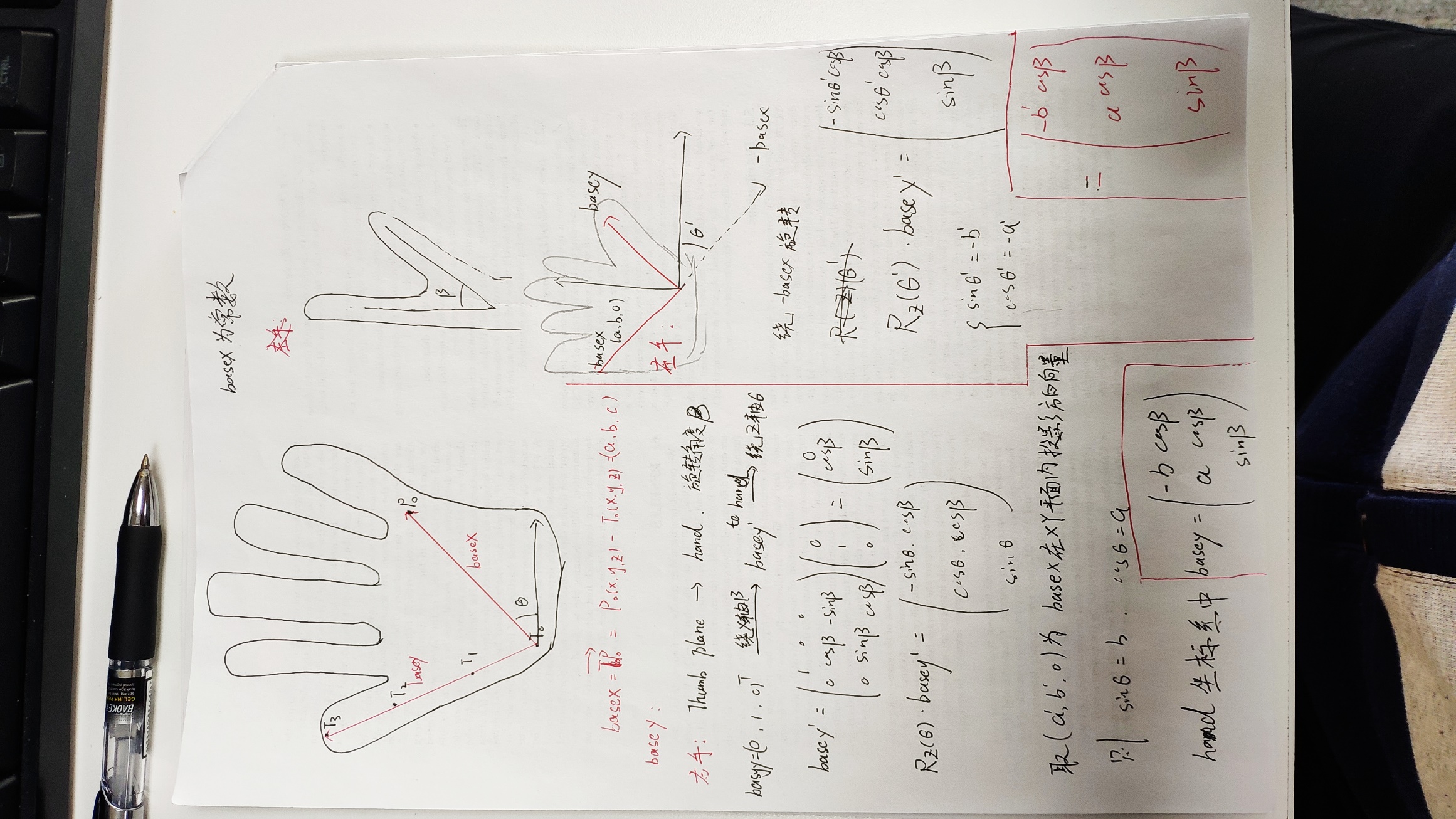
Handmodel为一个树状结构，手掌上六个点相对位置固定，即确定了手腕的SE3即可确定手掌上的六个点（手腕和每个手指的根节点），手掌参数设置见HandModel::LoadPalmParameters



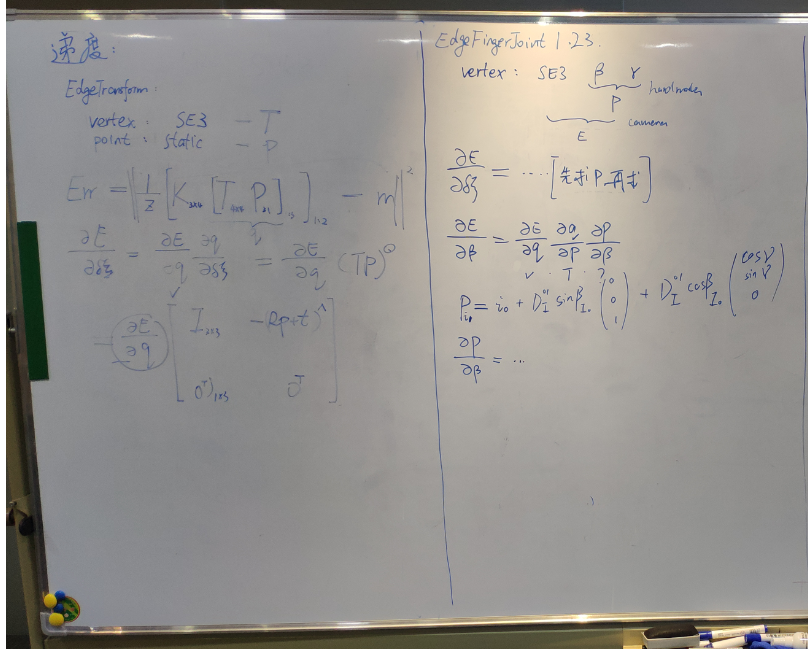
手指点的计算前提假设是手指的四个点在同一平面内，在hand坐标系内，除大拇指外的另外四根手指定义该平面的一组正交基向量为（0，0，1）和，为手指平面与XOY平面的相交线 和 （1，0，0）的夹角。所以在平面内，通过手指长度和弯曲角度，可表示1、2、3关节点为下图中的t1,t2,t3.计算详情见HandModel::UpdateKeypoints



大拇指定义与其他四指相似，只不过基向量选取不同，详见下图：



梯度上文中介绍了和，直接对t1、t2、t3中的公式求导即可。



#### 四、约束边

EdgeConstraint中每条边只有一个点（设计边时关联的点数越多，速度越慢）。EdgeConstraint的约束函数是分段函数，详见EdgeConstraint::computeError。 EdgeSmooth定义与 EdgeConstraint相似，只是EdgeSmooth的误差项为线性函数。

#### 五、Option

定义见aristo/skeleton/SkeletonOption.hpp，其中为ConsProjectPara为EdgeConsProject所使用的参数，其中para\_around, para\_devite, para\_far为三种惩罚系数，分别对应距离差小于threshold1，距离差在threshold1与threshold2之间，距离差大于threshold2。init\_dis为第一帧的初始值，当上一帧的值超出[lower\_bound, upper\_bound]的范围之后，重置上一帧的距离值为init\_dis。

OptimizerOption中定义的参数，iteration\_number为迭代优化的代数，weight系类参数为边的权重，一般用于信息矩阵的定义。Scale\_smooth 和 scale\_update\_limit与动态更新手掌大小相关，分别为步长和最大步长的限制。