

本文讲解g2o边的定义使用方法

小白：师兄，g2o框架《[从零开始一起学习SLAM | 理解图优化，一步步带你看懂g2o代码](#)》，以及顶点《[从零开始一起学习SLAM | 掌握g2o顶点编程套路](#)》我都学完啦，今天给我讲讲g2o中的边吧！是不是也有什么套路？

师兄：嗯，g2o的边比顶点稍微复杂一些，不过前面你也了解了许多g2o的东西，有没有发现g2o的编程基本都是固定的格式（套路）呢？

小白：是的，我现在按照师兄说的g2o框架和顶点设计方法，再去看g2o实现不同功能的代码，发现都是一个模子出来的，只不过在某些地方稍微改改就行了啊

师兄：是这样的。我们来看看g2o的边到底是咋回事。

初步认识g2o的边

师兄：在《g2o: A general Framework for (Hyper) Graph Optimization》这篇文档里，我们找到那张经典的类结构图，里面关于边（edge）的部分是这样的，重点是下图中红色框内。

上一次我们讲顶点的时候，还专门去追根溯源查找顶点类之间的继承关系，边其实也是类似的，我们在g2o官方GitHub上这些 `g2o/g2o/core/hyper_graph.h` `g2o/g2o/core/optimizable_graph.h` `g2o/g2o/core/base_edge.h`

头文件下就能看到这些继承关系了，我们就不像之前顶点那样一个个去追根溯源了，如果有兴趣你可以自己去试试看。我们主要关注一下上面红框内的三种边。

`BaseUnaryEdge`，`BaseBinaryEdge`，`BaseMultiEdge` 分别表示一元边，两元边，多元边。

小白：他们有啥区别啊？师兄：一元边你可以理解为一条边只连接一个顶点，两元边理解为一条边连接两个顶点，也就是我们常见的边啦，多元边理解为一条边可以连接多个（3个以上）顶点



一个比较丑的示例

下面我们来看看他们的参数有什么区别？你看主要就是几个参数：`D`, `E`, `VertexXi`, `VertexXj`，他们的分别代表：

`D` 是 `int` 型，表示测量值的维度（dimension）`E` 表示测量值的数据类型 `VertexXi`，`VertexXj` 分别表示不同顶点的类型

比如我们用边表示三维点投影到图像平面的重投影误差，就可以设置输入参数如下：

```
BaseBinaryEdge<2, Vector2D, VertexSBAPointXYZ, VertexSE3Expmap>
```

你说说看 这个定义是什么意思？小白：首先这个是个二元边。第1个2是说测量值是2维的，也就是图像像素坐标x,y的差值，对应测量值的类型是Vector2D，两个顶点也就是优化变量分别是三维点 VertexSBAPointXYZ，和李群位姿 VertexSE3Expmap？

师兄：对的，就是这样~当然除了输入参数外，定义边我们通常需要复写一些重要的成员函数 小白：听着和顶点类似哦，也是复写成员函数，顶点里主要复写了顶点更新函数oplusImpl和顶点重置函数setToOriginImpl，边的话是不是也差不多？师兄：边和顶点的成员函数还是差别比较大的，边主要有以下几个重要的成员函数

```
virtual bool read(std::istream& is);
virtual bool write(std::ostream& os) const;
virtual void computeError();
virtual void linearizeOplus();
```

下面简单解释一下 read, write：分别是读盘、存盘函数，一般情况下不需要进行读/写操作的话，仅仅声明一下就可以 computeError函数：非常重要，是使用当前顶点的值计算的测量值与真实的测量值之间的误差 linearizeOplus函数：非常重要，是在当前顶点的值下，该误差对优化变量的偏导数，也就是我们说的Jacobian

除了上面几个成员函数，还有几个重要的成员变量和函数也一并解释一下：

```
_measurement：存储观测值
_error：存储computeError() 函数计算的误差
_vertices[]：存储顶点信息，比如二元边的话，_vertices[] 的大小为2，存储顺序和调用setVertex(int, vertex) 是设定的int 有关（0 或1）
setId(int)：来定义边的编号（决定了在H矩阵中的位置）
setMeasurement(type) 函数来定义观测值
setVertex(int, vertex) 来定义顶点
setInformation() 来定义协方差矩阵的逆
```

后面我们写代码的时候经常会遇到他们的。

如何自定义g2o的边？

小白：前面你介绍了g2o中边的基本类型、重要的成员变量和成员函数，那么如果我们要定义边的话，具体如何编程呢？师兄：我这里正好有个模板给你看看，基本上定义g2o中的边，就是如下套路：

```
class myEdge: public g2o::BaseBinaryEdge<errorDim, errorType, Vertex1Type, Vertex2Type>
{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW
    myEdge(){}
    virtual bool read(istream& in) {}
    virtual bool write(ostream& out) const {}
    virtual void computeError() override
    {
        // ...
        _error = _measurement - Something;
    }
    virtual void linearizeOplus() override
```

```

{
    _jacobianOplusXi(pos, pos) = something;
    // ...
    /*
    _jacobianOplusXj(pos, pos) = something;
    ...
    */
}
private:
// data
}

```

我们可以发现，最重要的就是computeError(), linearizeOplus()两个函数了

小白：嗯，看起来好像也不难啊 师兄：我们先来看一个简单例子，地址在 https://github.com/gaoxiang12/slambook/blob/master/ch6/g2o_curve_fitting/main.cpp 这个是个一元边，主要是定义误差函数了，如下所示，你可以发现这个例子基本就是上面例子的一丢丢扩展，是不是感觉so easy？

```

// 误差模型 模板参数：观测值维度，类型，连接顶点类型
class CurveFittingEdge: public g2o::BaseUnaryEdge<1, double, CurveFittingVertex>
{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW
    CurveFittingEdge( double x ): BaseUnaryEdge(), _x(x) {}
    // 计算曲线模型误差
    void computeError()
    {
        const CurveFittingVertex* v = static_cast<const CurveFittingVertex*>
(_vertices[0]);
        const Eigen::Vector3d abc = v->estimate();
        _error(0,0) = _measurement - std::exp( abc(0,0)*_x*_x + abc(1,0)*_x + abc(2,0) );
    }
    virtual bool read( istream& in ) {}
    virtual bool write( ostream& out ) const {}
public:
    double _x; // x 值, y 值为 _measurement
};

```

小白：嗯，这个能看懂 师兄：下面是一个复杂一点例子，3D-2D点的PnP 问题，也就是最小化重投影误差问题，这个问题非常常见，使用最常见的二元边，弄懂了这个基本跟边相关的代码也差不多都一通百通了

代码在g2o的GitHub上这个地方可以看到 [g2o/types/sba/types_six_dof_expmap.h](#) 这里根据自己理解对代码加了注释，方便理解

```

//继承了BaseBinaryEdge类，观测值是2维，类型Vector2D,顶点分别是三维点、李群位姿
class G2O_TYPES_SBA_API EdgeProjectXYZ2UV : public BaseBinaryEdge<2, Vector2D,
VertexSBAPointXYZ, VertexSE3Expmap>{
public:
    EIGEN_MAKE_ALIGNED_OPERATOR_NEW;
    //1. 默认初始化
    EdgeProjectXYZ2UV();

```

```

//2. 计算误差
void computeError() {
    //李群相机位姿v1
    const VertexSE3Expmap* v1 = static_cast<const VertexSE3Expmap*>(_vertices[1]);
    // 顶点v2
    const VertexSBAPointXYZ* v2 = static_cast<const VertexSBAPointXYZ*>(_vertices[0]);
    //相机参数
    const CameraParameters * cam
        = static_cast<const CameraParameters *>(parameter(0));
    //误差计算, 测量值减去估计值, 也就是重投影误差obs-cam
    //估计值计算方法是T*p, 得到相机坐标系下坐标, 然后在利用camera2pixel()函数得到像素坐标。
    Vector2D obs(_measurement);
    _error = obs - cam->cam_map(v1->estimate().map(v2->estimate()));
}
//3. 线性增量函数, 也就是雅克比矩阵J的计算方法
virtual void linearizeOplus();
//4. 相机参数
CameraParameters * _cam;
bool read(std::istream& is);
bool write(std::ostream& os) const;
};

```

有一个地方比较难理解

```
_error = obs - cam->cam_map(v1->estimate().map(v2->estimate()));
```

小白：我确实看不懂这一句。。 师兄：其实就是：误差 = 观测 - 投影

下面我给你捋捋思路。我们先来看看cam_map 函数，它的定义在 g2o/types/sba/types_six_dof_expmap.cpp
cam_map 函数功能是把相机坐标系下三维点（输入）用内参转换为图像坐标（输出），具体代码如下所示

```

Vector2 CameraParameters::cam_map(const Vector3 & trans_xyz) const {
    Vector2 proj = project2d(trans_xyz);
    Vector2 res;
    res[0] = proj[0]*focal_length + principle_point[0];
    res[1] = proj[1]*focal_length + principle_point[1];
    return res;
}

```

然后看 .map函数，它的功能是把世界坐标系下三维点变换到相机坐标系，函数在 g2o/types/sim3/sim3.h 具体定义是

```

Vector3 map (const Vector3& xyz) const {
    return s*(r*xyz) + t;
}

```

因此下面这个代码

```
v1->estimate().map(v2->estimate())
```

就是用V1估计的pose把V2代表的三维点，变换到相机坐标系下。

小白：原来如此，以前我都忽视了这些东西了，没想到里面是这样的关联的。师兄：嗯，我们继续，前面主要是对 `computeError()` 的理解，还有一个很重要的函数就是 `linearizeOplus()`，用来定义雅克比矩阵 我摘取了相关代码（来自：`g2o/g2o/types/sba/types_six_dof_expmap.cpp`），并进行了标注，相信会更容易理解

十四讲第169页中的雅克比矩阵完全是按照书上 式子（7.45）、（7.47）来编程的，不难理解 小白：后面就是直接照抄书上就行，哈哈

如何向图中添加边？

师兄：前面我们讲过如何往图中增加顶点，可以说非常easy了，往图中增加边会稍微多一些内容，我们还是先从最简单的例子说起：一元边的添加方法

下面代码来自GitHub上，仍然是前面曲线拟合的例子 `slambook/ch6/g2o_curve_fitting/main.cpp`

```
// 往图中增加边
for ( int i=0; i<N; i++ )
{
    CurveFittingEdge* edge = new CurveFittingEdge( x_data[i] );
    edge->setId(i);
    edge->setVertex( 0, v );           // 设置连接的顶点
    edge->setMeasurement( y_data[i] ); // 观测数值
    edge->setInformation( Eigen::Matrix<double,1,1>::Identity()*1/(w_sigma*w_sigma) ); // 信息矩阵：协方差矩阵之逆
    optimizer.addEdge( edge );
}
```

小白：setMeasurement 函数的输入的观测值具体是指什么？师兄：对于这个曲线拟合，观测值就是实际观测到的数据点。对于视觉SLAM来说，通常就是我们观测到的特征点坐标，下面就是一个例子。这个例子比刚才的复杂一点，因为它是二元边，需要用边连接两个顶点 代码来自GitHub上 `slambook/ch7/pose_estimation_3d2d.cpp`

```
index = 1;
for ( const Point2f p:points_2d )
{
    g2o::EdgeProjectXYZ2UV* edge = new g2o::EdgeProjectXYZ2UV();
    edge->setId ( index );
    edge->setVertex ( 0, dynamic_cast<g2o::VertexSBAPointXYZ*>( optimizer.vertex ( index ) ) );
    edge->setVertex ( 1, pose );
    edge->setMeasurement ( Eigen::Vector2d ( p.x, p.y ) );
    edge->setParameterId ( 0,0 );
    edge->setInformation ( Eigen::Matrix2d::Identity() );
    optimizer.addEdge ( edge );
    index++;
}
```

小白：这里的setMeasurement函数里的p来自向量points_2d，也就是特征点的图像坐标(x,y)了吧！师兄：对，这正好呼应我刚才说的。另外，你看setVertex 有两个一个是 0 和 VertexSBAPointXYZ 类型的顶点，一个是1 和 pose。你觉得这里的0和1是什么意思？能否互换呢？

小白：0, 1应该是分别指代哪个顶点吧，直觉告诉我不能互换，可能得去查查顶点定义部分的代码 师兄：你的直觉没错！我帮你 查过啦，你看这个是setVertex在g2o官网的定义：

```
// set the ith vertex on the hyper-edge to the pointer supplied
void setVertex(size_t i, Vertex* v) { assert(i < _vertices.size() && "index out of bounds"); _vertices[i]=v;}
```

这段代码在 g2o/core/hyper_graph.h 里可以找到。你看 _vertices[i] 里的i就是我们这里的0和1，我们再去看看这里边的类型：g2o::EdgeProjectXYZ2UV 的定义，前面我们也放出来了，就这两句

```
class G2O_TYPES_SBA_API EdgeProjectXYZ2UV
{
    ....
    //李群相机位姿v1
    const VertexSE3Expmap* v1 = static_cast<const VertexSE3Expmap*>(_vertices[1]);
    // 顶点v2
    const VertexSBAPointXYZ* v2 = static_cast<const VertexSBAPointXYZ*>(_vertices[0]);
}
```

你看 vertices[0] 对应的是 VertexSBAPointXYZ 类型的顶点，也就是三维点，vertices[1] 对应的是 VertexSE3Expmap 类型的顶点，也就是位姿pose。因此前面 1 对应的就应该是 pose，0对应的 应该就是三维点。

小白：原来如此，之前都没注意这些，看来g2o不会帮我区分顶点的类型啊，以后这里编程要对应好，不然错了都找不到原因呢！谢谢师兄，今天又是收获满满的一天！

练习

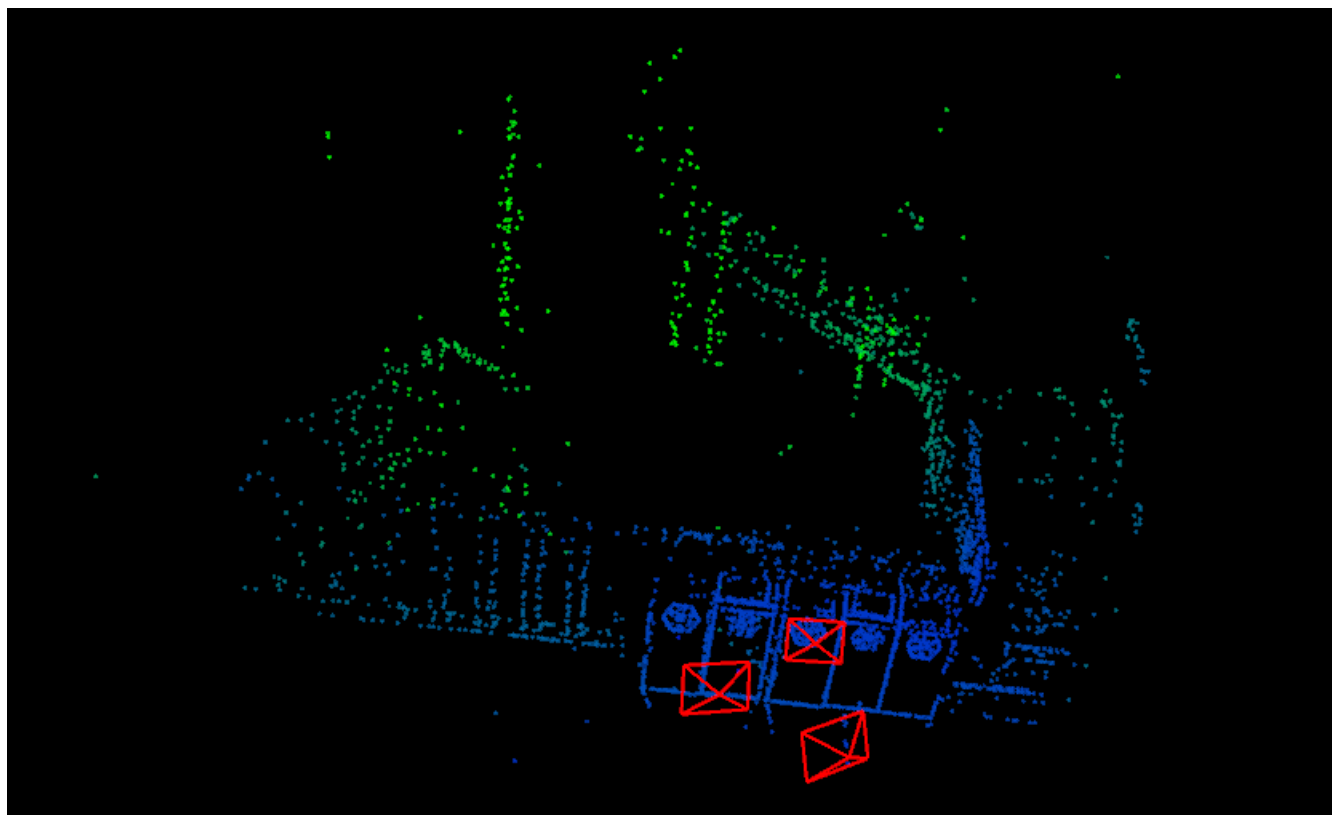
题目：用直接法Bundle Adjustment 估计相机位姿。给定3张图片，两个txt文件，其中poses.txt中存储3张图片对应的相机初始位姿（Tcw），格式为：timestamp, tx, ty, tz, qx, qy, qz, qw，分别对应时间戳、平移、旋转（四元数），而points.txt中存储的是3D点集合以及该点周围 4x4 窗口的灰度值，记做 $I(p)_i$ ，格式为：

x, y, z, 灰度1, 灰度2..., 灰度16

我们把每个3D点投影到对应图像中，用投影后点周围的灰度值与原始窗口的灰度值差异作为待优化误差。

请使用g2o进行优化，并绘制结果（绘制函数已经写好）。

代码框架中需要你填写顶点、边的定义。如果正确，输出结果如下图所示：



参考：

高翔《视觉 SLAM十四讲》 https://blog.csdn.net/try_again_later/article/details/81813639

代码框架、数据、窗口值具体顺序、优化目标函数、预期输出结果已经为你准备好了，公众号「计算机视觉life」后台回复：**边**，即可获得。

欢迎留言讨论，更多学习视频、文档资料、参考答案等关注计算机视觉life公众号，，菜单栏点击“知识星球”查看「从零开始学习SLAM」星球介绍，快来和其他小伙伴一起学习交流~



从零开始学习SLAM

自制视频课程
批改讲解作业
交流互动答疑
越早进越优惠



微信扫一扫

推荐阅读

[从零开始一起学习SLAM | 为什么要学SLAM?](#) [从零开始一起学习SLAM | 学习SLAM到底需要学什么?](#) [从零开始一起学习SLAM | SLAM有什么用?](#) [从零开始一起学习SLAM | C++新特性要不要学?](#) [从零开始一起学习SLAM | 为什么要用齐次坐标?](#) [从零开始一起学习SLAM | 三维空间刚体的旋转](#) [从零开始一起学习SLAM | 为啥需要李群与李代数?](#) [从零开始一起学习SLAM | 相机成像模型](#) [从零开始一起学习SLAM | 不推公式，如何真正理解对极约束?](#) [从零开始一起学习SLAM | 神奇的单应矩阵](#) [从零开始一起学习SLAM | 你好，点云](#) [从零开始一起学习SLAM | 给点云加个滤网](#) [从零开始一起学习SLAM | 点云平滑法线估计](#) [从零开始一起学习SLAM | 点云到网格的进化](#) [从零开始一起学习SLAM | 理解图优化，一步步带你看懂g2o代码](#) [从零开始一起学习SLAM | 掌握g2o顶点编程套路](#) [零基础小白，如何入门计算机视觉？](#) [SLAM领域牛人、牛实验室、牛研究成果梳理](#) [我用MATLAB撸了一个2D LiDAR SLAM](#) [可视化理解四元数，愿你不再掉头发](#) [最近一年语义SLAM有哪些代表性工作？](#) [视觉SLAM技术综述 汇总](#) [VIO、激光SLAM相关论文分类集锦](#) [研究SLAM，对编程的要求有多高？](#) [2018年SLAM、三维视觉方向求职经验分享](#) [深度学习遇到SLAM | 如何评价基于深度学习的DeepVO，VINet，VidLoc？](#)