逐行手写求解器

求解器的构造：

1. SetOrdering()

该函数主要用于对各个顶点进行排序，也就是它们出现在信息矩阵（H 矩阵）中的位置（包含了维数信息），首先明确以下几点：

a. 程序中的信息矩阵有两个:一个小的信息矩阵跟误差维数有关，比如用逆深度参数化后的视觉残差是两维的，则每一条边的信息矩阵的维度为 2*2，含有信息矩阵的残差从

r.transpose()*r

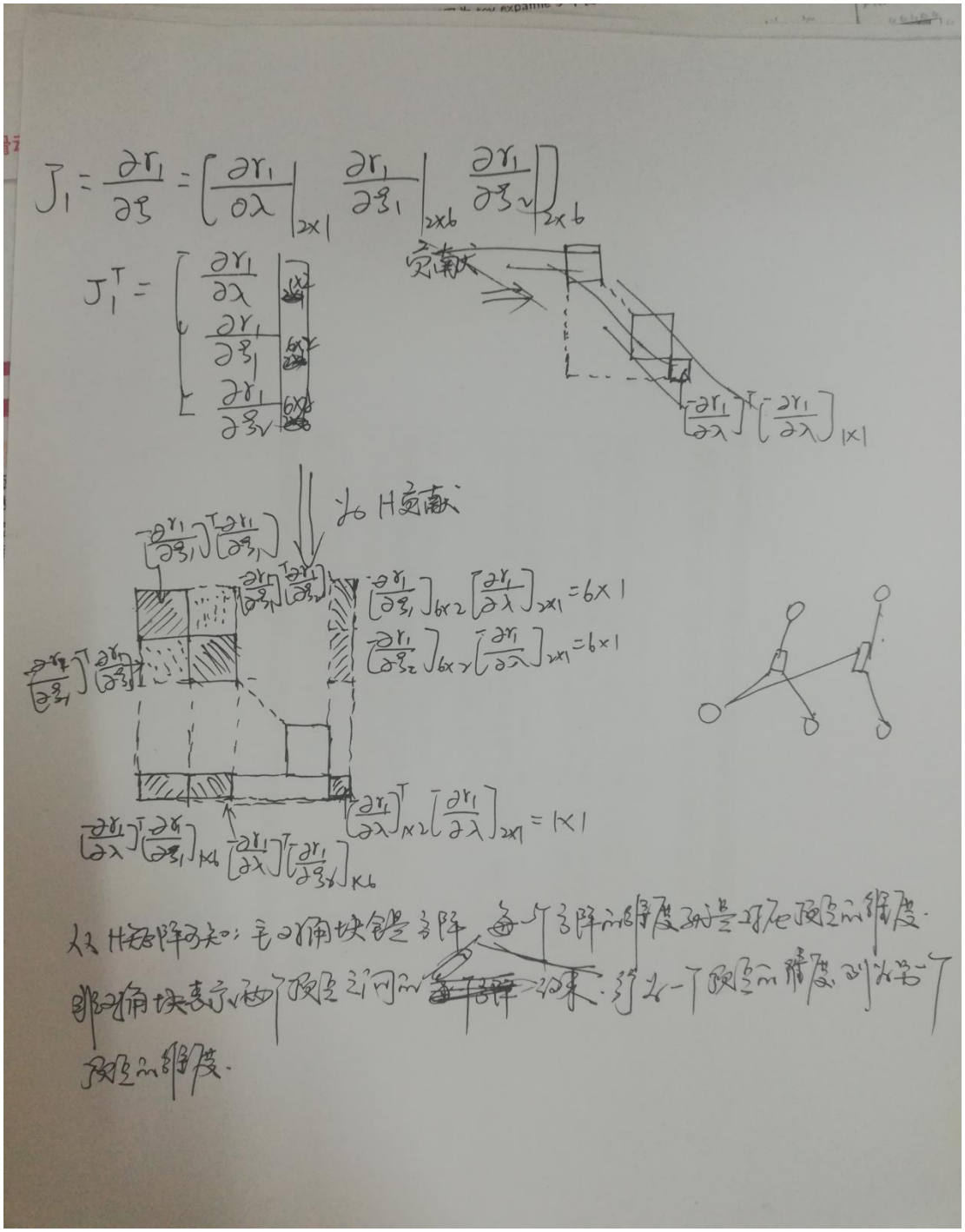
变为：

r.transpose()*sigma*r

另一个大的信息矩阵是由一个个小的信息矩阵拼接而成。其总的维数是：

顶点 1 的维数×顶点 1 的个数+顶点 2 的维数×顶点 2 的个数+……

b. 每一条边为信息矩阵 H 贡献一部分，以一条三元边（2 维）为例，顶点依次是：逆深度（1 维），相机 1 的位姿（6 维），相机 2 的位姿（6 维），把相机位姿排在前，逆深度排在后：



因此需要设置每一个顶点的位置，第一个逆深度块的位置应该是 0+所有相机位姿顶点的维数。第一个相机位姿顶点贡献的信息矩阵的位置应该在（0,0）处。以后每一个顶点贡献的信息矩阵块的位置都是在前一个同类型顶点的位置处加上顶点维数。假设有 3 个位姿顶点和 10 个路标点，则 H 矩阵的维度为

3*6+10*1=28 维

相机顶点在信息矩阵中的位置分别为：（0,0），（6,6），（12,12）

路标顶点(逆深度)在信息矩阵中的位置分别为 :(18,18),(19,19)...(27,27)

这就是排序的意义。

2. MakeHessian()

构造 H 矩阵，每一个小 H 矩阵的拼接

3. ComputeLambdaInitLM()

LM 初始化

4. SolveLinearSystem()

解线性方程，舒尔补求解，先 marg 掉路标点，得到对应的舒尔布，采用 SVD 等方式求得相机位姿的变化方向，再求路标点的变化方向。这里注意，对 H 矩阵求舒尔补，也要相应地变换 b 矩阵

5. UpdateStates()

更新状态量，只是更新了 delta_x，在此次试验中并没有使用先验信息与先验信息的传递（error_prior_和 H_prior，b_prior）。

6. IsGoodStepInLM()

判断当前步是否可行以及 LM 的 lambda 怎么更新

7. 如果是一次好的迭代，则在新的线性化点上重新建立 H 矩阵：

MakeHessian()

8. 关于 Eigen 库的语法总结：

A. 旋转向量对旋转矩阵赋值，注意 UnitZ 等单位向量的使用：

```
R = Eigen::AngleAxisd(theta, Eigen::Vector3d::UnitZ());
```

B. C++11 的真随机数：

```
#include <random>
std::default_random_engine generator;//生成器
std::normal_distribution<double> noise_gauss(0,0.001);//高斯分布器
double a=noise_gauss(generator);//生成高斯分布随机数
std::uniform_real_distribution<double> noise_uniform(-4.0,4.0);
Double b=noise_uniform(generator);//生成均匀分布随机数
```

C. 动态矩阵：

```
typedef Eigen::Matrix<double, Eigen::Dynamic, Eigen::Dynamic> MatXX;
typedef Eigen::Matrix<double, Eigen::Dynamic, 1> VecX;
```

使用时确定维度：

```
MatXX H(MatXX::Zero(size, size));//确定维度并初始化为 0
VecX b(VecX::Zero(size));
NatXX H(row_num,col_num);//确定维度
```

D. Eigen 矩阵的截取：

```
H.block(a,b,c,d);//从(a,b)开始截取 c 行 d 列
H.block(index_i,index_j,dim_i,dim_j).noalias() += hessian;//左右同事时出现同一个矩阵，如 H+=b，H*=b 时，为防止歧义，使用 noalias()
b.segment(a,b);//从 a 行开始截取 b 列
b.segment(index_i, dim_i).noalias() -= JtW *edge.second->Residual();
H.topLeftCorner(a,b);//截取左上角 a 行 b 列
H.topRightCorner(a,b);//右上角 a 行 b 列
H.bottomLeftCorner(a,b);
H.bottomRightCorner(a,b);
H.row(3);//截取第 2 行,注意区分 H.rows()返回行数
H.col(3);//截取第 2 列,注意区分 H.cols()返回列数
b.head(n);//前 n 行
b.tail(n);//后 n 行
```

E.关于 assert

```
//头文件:
#include <assert.h>
```

当 CMakeLists.txt 设置为

```
set(CMAKE_BUILD_TYPE DEBUG)
```

在编译期间会有效,并直接定位到错误;运行期间也会报错

如果要禁止 assert 作用,有两种方法:

1.头文件前添加#define NDEBUG

```
//#define NDEBUG
#include <assert.h>
```

2. CMakeLists.txt 设置为:

```
set(CMAKE_BUILD_TYPE DEBUG)
```