RootBA论文复现

1.计算Jacobian

```
//  TODO: task 1 calculate jacobian
Eigen::Matrix<Scalar, 3, 6> tempJp = Eigen::Matrix<Scalar, 3, 6>::Zero();
if (camera->IsFixed() == false)
{
    tempJp.template block<3, 3>(0, 0) = -R_cb * R_bw;
    tempJp.template block<3, 3>(0, 3) = R_cb * SkewSymmetricMatrix(p_b);
}

// 计算雅可比矩阵 d(相机坐标位置误差) / d(特征点 3 自由度位置)
Eigen::Matrix<Scalar, 3, 3> tempJl = Eigen::Matrix<Scalar, 3, 3>::Zero();
if (this->landmark->IsFixed() == false)
{
    tempJl = R_cb * R_bw;
}

// 计算雅可比矩阵 d(相机坐标位置误差) / d(相机外参 6 自由度位姿)
// Hint: to calculate skew symmetric matrix,
// you could use function SkewSymmetricMatrix(*)
Eigen::Matrix<Scalar, 3, 6> tempJex = Eigen::Matrix<Scalar, 3, 6>::Zero();
if (this->exPose->IsFixed() == false)
{
    tempJex.template block<3, 3>(0, 0) = -R_cb;
    tempJex.template block<3, 3>(0, 3) = SkewSymmetricMatrix(p_c);
}
// Fill in above this line
```

2.执行QR分解，并更新storage矩阵

```
// TODO: task 2 执行QR分解，并更新sorage矩阵
// 对 Jl 进行 QR 分解
// HInt: Eigen::HouseholderQR could be used to perform QR decompose.
Eigen::HouseholderQR<MatrixX<Scalar>> qr;
qr.compute(Jl);
MatrixX<Scalar> Q = qr.householderQ();

// // 对 storage 矩阵左乘 Q.T
this->storage.block(0, cols - 4, rows - 3, 3) = Q.transpose() * Jl;
this->storage.block(0, 0, rows - 3, cols - 4) = Q.transpose() * Jp;
this->storage.block(0, cols - 1, rows - 3, 1) = Q.transpose() * r;

// Fill in above this line
```

3.进行givens旋转，让storage矩阵指定位置数值为0，并存储中间结果

```
// TODO: task 3 进行givens旋转，让storage矩阵指定位置数值为0,并存储中间结果
// Hint: makeGivens()函数可以进行givens旋转，
// 其方法本质上就是传入 x1 和 x2，计算出旋转矩阵对应的 sin 和 cos
gr.makeGivens(storage(i, n), storage(m, n));

// 对 this->storage 矩阵进行初等变换，并将连乘结果记录下来
this->storage.applyOnTheLeft(i, m, gr.adjoint());
this->Q_lambda.applyOnTheLeft(i, m, gr.adjoint());

// Fill in above this line
```

4.对storage矩阵实现状态回退

```
//  TODO: task 3 对storage矩阵实现状态回退
this->storage = this->Q_lambda.transpose() * this->storage;

// Fill in above this line
```

代码运行效果为：

```
xwl@xwl-Inspiron-15-7000-Gaming:~/Documents/VSLAM-fundamentals-and-VIO-learning/L18-vio-project/Project BASolver/rootBA/bin$ ./benchmark_rootba
Test rootBA solver on Mono BA problem.

Step 1: Prepare dataset.
iter      cost      cost_change  |gradient|   |step|    tr_ratio  tr_radius  ls_iter  iter_time  total_time
   0  7.145302e+01    0.00e+00    3.07e+01   0.00e+00   0.00e+00  1.00e+04        0    3.15e-03    9.98e-03
   1  1.098216e+00    7.04e+01    3.28e+00   1.12e+01   9.86e-01  3.00e+04        1    5.50e-03    1.59e-02
   2  7.684180e-03    1.09e+00    1.88e+00   2.26e+00   9.96e-01  9.00e+04        1    3.85e-03    1.98e-02
   3  2.542021e-03    5.14e-03    1.04e-02   2.19e-01   1.00e+00  2.70e+05        1    3.31e-03    2.31e-02
   4  2.541646e-03    3.75e-07    6.27e-06   3.28e-03   1.00e+00  8.10e+05        1    3.36e-03    2.65e-02
<ceres> cost time 29.8912

Step 2: Construct problem solver.
problem has 1 ex pose.

Step 3: Add camera pose vertices.
problem has 10 cameras.

Step 4: Add landmark position vertices.
problem has 300 landmarks.

Step 5: Show the initial value of parameters.
================== Initial parameters -> Landmark Position ==================
  id 0 : gt [ -2.24833 -0.330799   4.52615],    op [ -1.90618 -0.874208   4.46517]
  id 1 : gt [-3.51549  3.18125  4.66603],       op [-3.17334  2.63784  4.60505]
  id 2 : gt [-0.012159  -3.73557   6.28662],    op [0.329986 -4.27898  6.22564]
  id 3 : gt [   -2.374 -0.188546   7.21229],    op [ -2.03185 -0.731955   7.15131]
  id 4 : gt [1.83798 3.01253 5.65611],  op [2.18013 2.46912 5.59513]
  id 5 : gt [-2.29163  1.03657  4.32695],       op [-1.94948 0.493166  4.26597]
  id 6 : gt [-2.7065 1.62392 6.35196],  op [-2.36435  1.08051  6.29097]
  id 7 : gt [3.76069 3.34278 6.57192],  op [4.10284 2.79937 6.51093]
  id 8 : gt [-2.79751 0.159908  5.91122],       op [ -2.45536 -0.383501   5.85023]
  id 9 : gt [-3.35718  3.78677  7.34483],       op [-3.01503  3.24336  7.28384]
  id 10 : gt [ 1.94619 -1.36755  7.90872],      op [ 2.28834 -1.91096  7.84774]
================== Initial parameters -> Camera Rotation ==================
  id 0 : gt [1, 0, 0, 0],       op [1, 0, 0, 0]
  id 1 : gt [0.999807, 0, 0, 0.0196337],        op [0.999807, 0, 0, 0.0196337]
  id 2 : gt [0.999229, 0, 0, 0.0392598],        op [0.997326, 0.0491485, 0.041396, 0.034801]
  id 3 : gt [0.998266, 0, 0, 0.0588708],        op [0.998552, -0.0134883, -0.0399108, 0.0334417]
  id 4 : gt [0.996917, 0, 0, 0.0784591],        op [0.996599, -0.0444865, -0.00457502, 0.0692104]
  id 5 : gt [0.995185, 0, 0, 0.0980171],        op [0.987753, -0.0378763, 0.0679402, 0.135252]
  id 6 : gt [0.993068, 0, 0, 0.117537], op [0.992802, 0.0597063, 0.0102714, 0.103319]
  id 7 : gt [0.990569, 0, 0, 0.137012], op [0.984884, 0.0452931, 0.0905161, 0.140563]
  id 8 : gt [0.987688, 0, 0, 0.156434], op [0.990327, -0.013265, -0.0842032, 0.109479]
  id 9 : gt [0.984427, 0, 0, 0.175796], op [0.98495, 0.0377093, 0.0151174, 0.167995]
================== Initial parameters -> Camera Position ==================
  id 0 : gt [0 0 0],    op [0 0 0]
  id 1 : gt [-0.00616771    0.314079   0.0784591],      op [-0.00616771    0.314079   0.0784591]
  id 2 : gt [-0.0246613   0.627673   0.156434], op [ -0.21183  0.889357 0.0190035]
  id 3 : gt [-0.0554523   0.940299   0.233445], op [-0.0453788   0.537242  -0.301028]
  id 4 : gt [-0.0984933    1.25148   0.309017], op [-0.0231109    1.42312   0.294269]
  id 5 : gt [-0.153718   1.56072  0.382683],    op [-0.0192039    1.49368   0.232829]
  id 6 : gt [-0.221041   1.86756   0.45399],    op [-0.315232   1.84135  0.191135]
```

```
  id 7 : gt [-0.300358   2.17152  0.522499],    op [-0.363409    2.0904  0.536953]
  id 8 : gt [-0.391548   2.47214  0.587785],    op [-0.585898   2.20702  0.808726]
  id 9 : gt [-0.494469   2.76894  0.649448],    op [-0.893197   3.05257  0.963362]

Step 6: Start solve problem.
<Iter 0 / 100> cost 2.17501, dx 11.1552, lambda 1.56516e-06, rho 0.0866106, time cost 15.023 ms
<Iter 1 / 100> cost 0.0155108, dx 2.26583, lambda 2.23605e-06, rho 0.123005, time cost 6.79956 ms
<Iter 2 / 100> cost 0.00509086, dx 0.18585, lambda 2.80157e-06, rho 0.183803, time cost 6.8584 ms
<Iter 3 / 100> cost 0.00509, dx 0.00332012, lambda 4.20771e-06, rho 0.102644, time cost 5.90439 ms
<Iter 4 / 100> cost 0.00509, dx 0.000108079, lambda 8.41542e-06, rho -0.00139561, time cost 8.28589 ms
<End> min cost holding times up to threshold, finished.
<Finish> Problem solve totally cost 50.1015 ms

Step 7: Compare optimization result with ground truth.
================== Solve result -> Landmark Position ==================
  id 0 : gt [ -2.24833 -0.330799   4.52615],    op [ -2.24528 -0.329491   4.51781]
  id 1 : gt [-3.51549  3.18125  4.66603],       op [-3.51526  3.17818  4.66203]
  id 2 : gt [-0.012159  -3.73557   6.28662],    op [-0.0113791   -3.74182     6.2919]
  id 3 : gt [   -2.374 -0.188546   7.21229],    op [ -2.37073 -0.190228   7.20641]
  id 4 : gt [1.83798 3.01253 5.65611],  op [1.83402 3.00814 5.64486]
  id 5 : gt [-2.29163  1.03657  4.32695],       op [-2.29024  1.03541  4.31999]
  id 6 : gt [-2.7065 1.62392 6.35196],  op [-2.70503   1.6207  6.34438]
  id 7 : gt [3.76069 3.34278 6.57192],  op [3.74657  3.3345 6.54721]
  id 8 : gt [-2.79751 0.159908  5.91122],       op [-2.79203 0.161492  5.89855]
  id 9 : gt [-3.35718  3.78677  7.34483],       op [-3.35006  3.77755  7.32459]
  id 10 : gt [ 1.94619 -1.36755  7.90872],      op [ 1.94811 -1.37412  7.91321]
================== Solve result -> Camera Rotation ==================
  id 0 : gt [1, 0, 0, 0],       op [1, 0, 0, 0]
  id 1 : gt [0.999807, 0, 0, 0.0196337],        op [0.999807, 0, 0, 0.0196337]
  id 2 : gt [0.999229, 0, 0, 0.0392598],        op [0.999227, -6.69981e-05, 0.000180589, 0.0393232]
  id 3 : gt [0.998266, 0, 0, 0.0588708],        op [0.998263, -1.97011e-05, 0.000129779, 0.0589124]
  id 4 : gt [0.996917, 0, 0, 0.0784591],        op [0.996907, -2.25279e-05, 0.000181526, 0.0785918]
  id 5 : gt [0.995185, 0, 0, 0.0980171],        op [0.995179, -1.90173e-05, 5.11822e-05, 0.0980782]
  id 6 : gt [0.993068, 0, 0, 0.117537], op [0.99306, 5.34146e-05, 7.05122e-05, 0.117611]
  id 7 : gt [0.990569, 0, 0, 0.137012], op [0.990563, -0.000120062, 8.99134e-05, 0.137058]
  id 8 : gt [0.987688, 0, 0, 0.156434], op [0.987666, -6.89661e-05, 0.000216976, 0.156577]
  id 9 : gt [0.984427, 0, 0, 0.175796], op [0.984406, 3.42207e-07, 9.80555e-05, 0.175913]
================== Solve result -> Camera Position ==================
  id 0 : gt [0 0 0],    op [0 0 0]
  id 1 : gt [-0.00616771    0.314079   0.0784591],      op [-0.00616771    0.314079   0.0784591]
  id 2 : gt [-0.0246613   0.627673   0.156434], op [-0.0277977   0.625625   0.155064]
  id 3 : gt [-0.0554523   0.940299   0.233445], op [-0.0575513   0.939318   0.231988]
  id 4 : gt [-0.0984933    1.25148   0.309017], op [-0.101487    1.2487  0.308838]
  id 5 : gt [-0.153718   1.56072  0.382683],    op [-0.154589   1.55846  0.380405]
  id 6 : gt [-0.221041   1.86756   0.45399],    op [-0.222379   1.86522  0.452486]
  id 7 : gt [-0.300358   2.17152  0.522499],    op [-0.301693   2.16697  0.519713]
  id 8 : gt [-0.391548   2.47214  0.587785],    op [-0.394571   2.46792  0.585021]
  id 9 : gt [-0.494469   2.76894  0.649448],    op [-0.496511   2.76468  0.647536]
```

```
Step 8: Compute average residual.
  Average landmark position residual for initialization is 0.645039 m
  Average landmark position residual for rootBA is      0.0139406 m
  Average landmark position residual for CERES is       0.0641895 m
  Average camera atitude residual for RootBA is         0.000117882 rad, 0.00675411 deg
  Average camera atitude residual for Ceres is  0.000442923 rad, 0.0253776 deg
  Average camera atitude residual at initialization is 0.109683 rad, 6.28435 deg
  Average camera position residual for RootBA is        0.00336946 m
  Average camera position residual for Ceres is         0.0118743 m
  Average camera position residual for initialization is 0.278127 m
xwl@xwl-Inspiron-15-7000-Gaming:~/Documents/VSLAM-fundamentals-and-VIO-learning/L18-vio-project/Project BASolver/rootBA/bin$
```