深蓝学院
shenlanxueyuan.com

第五次作业思路

主讲人 张松鹏

公式推导

残差

残差：

$$r_a = \|\boldsymbol{g}\|^2 - \|\boldsymbol{a}\|^2$$

其中

$$\boldsymbol{a} = (\boldsymbol{I} - \boldsymbol{S}_a)\,\boldsymbol{K}_a'\,(\boldsymbol{A} - \boldsymbol{b}_a)$$

$$\boldsymbol{S}_a = \begin{bmatrix} 0 & 0 & 0 \\ S_{ayx} & 0 & 0 \\ S_{azx} & S_{azy} & 0 \end{bmatrix} \quad \boldsymbol{K}_a' = \begin{bmatrix} K_{ax}' & 0 & 0 \\ 0 & K_{ay}' & 0 \\ 0 & 0 & K_{az}' \end{bmatrix} \quad \boldsymbol{b}_a = \begin{bmatrix} b_{ax} \\ b_{ay} \\ b_{az} \end{bmatrix}$$

公式推导

●残差

加速度向量可展开为

$$\boldsymbol{a}=\begin{bmatrix}1 & 0 & 0\\ -S_{ayx} & 1 & 0\\ -S_{azx} & -S_{azy} & 1\end{bmatrix}\begin{bmatrix}K'_{ax} & 0 & 0\\ 0 & K'_{ay} & 0\\ 0 & 0 & K'_{az}\end{bmatrix}\begin{bmatrix}A_x-b_{ax}\\ A_y-b_{ay}\\ A_z-b_{az}\end{bmatrix}$$

$$=\begin{bmatrix}K'_{ax} & 0 & 0\\ -S_{ayx}K'_{ax} & K'_{ay} & 0\\ -S_{azx}K'_{ax} & -S_{azy}K'_{ay} & K'_{az}\end{bmatrix}\begin{bmatrix}A_x-b_{ax}\\ A_y-b_{ay}\\ A_z-b_{az}\end{bmatrix}$$

$$=\begin{bmatrix}K'_{ax}\left(A_x-b_{ax}\right)\\ -S_{ayx}K'_{ax}\left(A_x-b_{ax}\right)+K'_{ay}\left(A_y-b_{ay}\right)\\ -S_{azx}K'_{ax}\left(A_x-b_{ax}\right)-S_{azy}K'_{ay}\left(A_y-b_{ay}\right)+K_{az}\left(A_z-b_{az}\right)\end{bmatrix}$$

待估计参数为

$$\boldsymbol{\theta}=\left[S_{ayx},S_{azx},S_{azy},K'_{ax},K'_{ay},K'_{az},b_{ax},b_{ay},b_{az}\right]^T$$

公式推导

深蓝学院 shenlanxueyuan.com

- 雅可比

根据链式求导分解为

$$\frac{dr_a}{d\boldsymbol{\theta}} = \frac{dr_a}{d\boldsymbol{a}}\frac{d\boldsymbol{a}}{d\boldsymbol{\theta}}$$

其中

$$\frac{dr_a}{d\boldsymbol{a}} = -2\boldsymbol{a}^T$$

$$\frac{d\boldsymbol{a}}{d\boldsymbol{\theta}_{123}} = \begin{bmatrix} 0 & 0 & 0 \\ -K'_{ax}\left(A_x - b_{ax}\right) & 0 & 0 \\ 0 & -K'_{ax}\left(A_x - b_{ax}\right) & -K'_{ay}\left(A_y - b_{ay}\right) \end{bmatrix}$$

$$\frac{d\boldsymbol{a}}{d\boldsymbol{\theta}_{456}} = \begin{bmatrix} \left(A_x - b_{ax}\right) & 0 & 0 \\ -S_{ayx}\left(A_x - b_{ax}\right) & \left(A_y - b_{ay}\right) & 0 \\ -S_{azx}\left(A_x - b_{ax}\right) & -S_{azy}\left(A_y - b_{ay}\right) & \left(A_z - b_{az}\right) \end{bmatrix}$$

$$\frac{d\boldsymbol{a}}{d\boldsymbol{\theta}_{789}} = \begin{bmatrix} -K'_{ax} & 0 & 0 \\ S_{ayx}K'_{ax} & -K'_{ay} & 0 \\ S_{azx}K'_{ax} & S_{azy}K'_{ay} & -K_{az} \end{bmatrix}$$

代码实现

●自动求导

只需要修改以下3个地方，将加速度计内参模型改为下三角：

1) MultiPosAccResidual中的operator函数

```
CalibratedTriad_<_T2> calib_triad(
  //
  // TODO: implement lower triad model here
  //
  // mis_yz, mis_zy, mis_zx:
  _T2(0), _T2(0), _T2(0),
  // mis_xz, mis_xy, mis_yx:
  params[0], params[1], params[2],
  //    s_x,     s_y,     s_z:
  params[3], params[4], params[5],
  //    b_x,     b_y,     b_z:
  params[6], params[7], params[8]
);
```

代码实现

- 自动求导

2) MultiPosCalibration_中的calibrateAcc函数部分1

```
//
// TODO: implement lower triad model here
//
acc_calib_params[0] = init_acc_calib_.misXZ();
acc_calib_params[1] = init_acc_calib_.misXY();
acc_calib_params[2] = init_acc_calib_.misYX();

acc_calib_params[3] = init_acc_calib_.scaleX();
acc_calib_params[4] = init_acc_calib_.scaleY();
acc_calib_params[5] = init_acc_calib_.scaleZ();

acc_calib_params[6] = init_acc_calib_.biasX();
acc_calib_params[7] = init_acc_calib_.biasY();
acc_calib_params[8] = init_acc_calib_.biasZ();
```

代码实现

- 自动求导

3) MultiPosCalibration_中的calibrateAcc函数部分2

```
acc_calib_ = CalibratedTriad_<_T>(
    //
    // TODO: implement lower triad model here
    //
    0,0,0,
    min_cost_calib_params[0],
    min_cost_calib_params[1],
    min_cost_calib_params[2],
    min_cost_calib_params[3],
    min_cost_calib_params[4],
    min_cost_calib_params[5],
    min_cost_calib_params[6],
    min_cost_calib_params[7],
    min_cost_calib_params[8]
);
```

代码实现

●解析求导

修改优化参数与内参矩阵对应关系

```
template <typename _T>
  imu_tk::CalibratedTriad_<_T>::CalibratedTriad_( const _T &mis_yz, const _T &mis_zy, const _T &mis_zx,
                                                 const _T &mis_xz, const _T &mis_xy, const _T &mis_yx,
                                                 const _T &s_x, const _T &s_y, const _T &s_z,
                                                 const _T &b_x, const _T &b_y, const _T &b_z )
{
  mis_mat_ <<  _T(1)   , -mis_yz  ,  -mis_zy  ,
              -mis_xz ,  _T(1)   , -mis_zx  ,
             -mis_xy ,  -mis_yx   ,  _T(1)   ;
```

```
inline _T misYZ() const { return -mis_mat_(0,1); };
inline _T misZY() const { return -mis_mat_(0,2); };
inline _T misZX() const { return -mis_mat_(1,2); };
inline _T misXZ() const { return -mis_mat_(1,0); };
inline _T misXY() const { return -mis_mat_(2,0); };
inline _T misYX() const { return -mis_mat_(2,1); };
```

代码实现

● 解析求导

自定义MultiPosAccResidual类继承自ceres::SizedCostFunction<1, 9>

```
template <typename _T1>
class MultiPosAccResidual : public ceres::SizedCostFunction<1, 9>
{
public:
  MultiPosAccResidual(const _T1 &g_mag,
                      const Eigen::Matrix<_T1, 3, 1> &sample) : g_mag_(g_mag),
                                                                sample_(sample) {}
  virtual ~MultiPosAccResidual() {}
  virtual bool Evaluate(double const *const *params, double *residuals, double **jacobians) const
  {
    Eigen::Matrix<double, 3, 1> raw_samp(double(sample_(0)), double(sample_(1)), double(sample_(2)));
    CalibratedTriad_<double> calib_triad(
        double(0), double(0), double(0),
        params[0][0], params[0][1], params[0][2],
        params[0][3], params[0][4], params[0][5],
        params[0][6], params[0][7], params[0][8]);
    Eigen::Matrix<double, 3, 1> calib_samp = calib_triad.unbiasNormalize(raw_samp);
    residuals[0] = double(g_mag_ * g_mag_) - calib_samp.squaredNorm();
```

代码实现

●解析求导

```
    if (jacobians != NULL)
    {
      if (jacobians[0] != NULL)
      {
        Eigen::Matrix<double, 1, 3> drda = -2 * calib_samp.transpose();
        Eigen::Matrix<double, 3, 9> dadtheta = Eigen::Matrix<double, 3, 9>::Zero();
        dadtheta(1, 0) = -params[0][3] * (raw_samp(0) - params[0][6]);
        dadtheta(2, 1) = dadtheta(1, 0);
        dadtheta(2, 2) = -params[0][4] * (raw_samp(1) - params[0][7]);
        dadtheta(0, 3) = raw_samp(0) - params[0][6];
        dadtheta(1, 3) = -params[0][0] * (raw_samp(0) - params[0][6]);
        dadtheta(2, 3) = -params[0][1] * (raw_samp(0) - params[0][6]);
        dadtheta(1, 4) = raw_samp(1) - params[0][7];
        dadtheta(2, 4) = -params[0][2] * (raw_samp(1) - params[0][7]);
        dadtheta(2, 5) = raw_samp(2) - params[0][8];
        dadtheta(0, 6) = -params[0][3];
        dadtheta(1, 6) = params[0][0] * params[0][3];
        dadtheta(2, 6) = params[0][1] * params[0][3];
        dadtheta(1, 7) = -params[0][4];
        dadtheta(2, 7) = params[0][2] * params[0][4];
        dadtheta(2, 8) = -params[0][5];

        Eigen::Map<Eigen::Matrix<double, 1, 9, Eigen::RowMajor>> J_se3(jacobians[0]);
        J_se3.setZero();
        J_se3 = drda * dadtheta;
      }
    }
    return true;
}
```

```
  static ceres::CostFunction* Create ( const _T1 &g_mag, const Eigen::Matrix< _T1, 3 , 1> &sample )
  {
    return ( new MultiPosAccResidual<_T1>( g_mag, sample ) );
  }

  const _T1 g_mag_;
  const Eigen::Matrix< _T1, 3 , 1> sample_;
};
```

在线问答

深蓝学院
shenlanxueyuan.com

Q&A

感谢各位聆听！

Thanks for Listening