深蓝学院
shenlanxueyuan.com

# 第三次作业思路

主讲人 张松鹏

# 公式推导



## ●线特征

残差：

$$d_{\mathcal{E}} = |\boldsymbol{d}_{\mathcal{E}}| = \sqrt{\boldsymbol{d}_{\mathcal{E}}^T \boldsymbol{d}_{\mathcal{E}}}$$

其中

$$\boldsymbol{d}_{\mathcal{E}} = \frac{(\tilde{p}_i - p_b) \times (\tilde{p}_i - p_a)}{|p_a - p_b|}$$

$$\tilde{p}_i = \boldsymbol{R} p_i + \boldsymbol{t}$$

根据链式求导法则，残差对位移和姿态的雅可比写为

$$\frac{\partial d_{\mathcal{E}}}{\partial \boldsymbol{T}} = \frac{\partial d_{\mathcal{E}}}{\partial \boldsymbol{d}_{\mathcal{E}}} \frac{\partial \boldsymbol{d}_{\mathcal{E}}}{\partial \tilde{p}_i} \frac{\partial \tilde{p}_i}{\partial \boldsymbol{T}}$$

其中 $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t} \\ \delta\boldsymbol{\theta} \end{bmatrix}$，旋转采用SO3左扰动模型，即 $\boldsymbol{R}' = \exp(\boldsymbol{\theta})\boldsymbol{R}$

# 公式推导



## ●线特征

对雅可比中的3个因子依次求解：

$$
\begin{aligned}
\frac{\partial d_\varepsilon}{\partial \boldsymbol{d}_\varepsilon} &= \frac{\partial \sqrt{\boldsymbol{d}_\varepsilon^T \boldsymbol{d}_\varepsilon}}{\partial \boldsymbol{d}_\varepsilon} \\
&= \frac{1}{2} \frac{1}{\sqrt{\boldsymbol{d}_\varepsilon^T \boldsymbol{d}_\varepsilon}} 2\boldsymbol{d}_\varepsilon^T \\
&= \frac{\boldsymbol{d}_\varepsilon^T}{d_\varepsilon}
\end{aligned}
$$

$$
\begin{aligned}
\frac{\partial \boldsymbol{d}_\varepsilon}{\partial \tilde{p}_i} &= \frac{\partial \frac{(\tilde{p}_i - p_b)\times(\tilde{p}_i - p_a)}{|p_a - p_b|}}{\partial \tilde{p}_i} \\
&= \frac{1}{|p_a - p_b|} \frac{\partial\left((\tilde{p}_i - p_b) \times (\tilde{p}_i - p_a)\right)}{\partial \tilde{p}_i} \\
&= \frac{1}{|p_a - p_b|} \frac{\partial\left(\tilde{p}_i - p_b\right)}{\partial \tilde{p}_i} \times (\tilde{p}_i - p_a) \\
&\quad + \frac{1}{|p_a - p_b|} (\tilde{p}_i - p_b) \times \frac{\partial\left(\tilde{p}_i - p_a\right)}{\partial \tilde{p}_i} \\
&= -\frac{1}{|p_a - p_b|} (\tilde{p}_i - p_a)_\times + \frac{1}{|p_a - p_b|} (\tilde{p}_i - p_b)_\times \\
&= \frac{1}{|p_a - p_b|} (p_a - p_b)_\times
\end{aligned}
$$

$$
\frac{\partial \tilde{p}_i}{\partial \boldsymbol{t}} = \boldsymbol{I}
$$

$$
\begin{aligned}
\frac{\partial \tilde{p}_i}{\partial \delta\boldsymbol{\theta}} &= \frac{\partial\left(\exp(\delta\boldsymbol{\theta})\boldsymbol{R}p_i + \boldsymbol{t}\right)}{\partial \delta\boldsymbol{\theta}} \\
&= \frac{\partial\left(\left(\boldsymbol{I} + \delta\boldsymbol{\theta}^\wedge\right)\boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}} \\
&= \frac{\partial\left(\delta\boldsymbol{\theta}^\wedge \boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}} \\
&= \frac{-\left(\boldsymbol{R}p_i\right)_\times \delta\boldsymbol{\theta}}{\partial \delta\boldsymbol{\theta}} \\
&= -\left(\boldsymbol{R}p_i\right)_\times
\end{aligned}
$$

# 公式推导



● 线特征

最后得到

$$\frac{\partial d_{\mathcal{E}}}{\partial \boldsymbol{t}} = \frac{\boldsymbol{d}_{\mathcal{E}}^{T}}{d_{\mathcal{E}}} \frac{1}{|p_a - p_b|} (p_a - p_b)_{\times}$$

$$\frac{\partial d_{\mathcal{E}}}{\partial \delta \boldsymbol{\theta}} = -\frac{\boldsymbol{d}_{\mathcal{E}}^{T}}{d_{\mathcal{E}}} \frac{1}{|p_a - p_b|} (p_a - p_b)_{\times} (\boldsymbol{R} p_i)_{\times}$$

# 公式推导

## ●面特征

残差：

$$d_{\mathcal{H}} = |\boldsymbol{d}_{\mathcal{H}}|$$

其中

$$\boldsymbol{d}_{\mathcal{H}} = (\tilde{p}_i - p_j) \cdot \frac{(p_l - p_j) \times (p_m - p_j)}{|(p_l - p_j) \times (p_m - p_j)|}$$

根据链式求导法则，残差对位移和姿态的雅可比写为

$$\frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{T}} = \frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{d}_{\mathcal{H}}} \frac{\partial \boldsymbol{d}_{\mathcal{H}}}{\partial \tilde{p}_i} \frac{\partial \tilde{p}_i}{\partial \boldsymbol{T}}$$

其中 $\boldsymbol{T} = \begin{bmatrix} \boldsymbol{t} \\ \delta\boldsymbol{\theta} \end{bmatrix}$，旋转采用SO3左扰动模型，即 $\boldsymbol{R}' = \exp(\boldsymbol{\theta})\boldsymbol{R}$

# 公式推导



● 面特征

对雅可比中的3个因子依次求解:

$$\frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{d}_{\mathcal{H}}}=\frac{\boldsymbol{d}_{\mathcal{H}}^{T}}{d_{\mathcal{H}}}$$

$$\frac{\partial \boldsymbol{d}_{\mathcal{H}}}{\partial \tilde{p}_i}=\frac{\left((p_l-p_j)\times(p_m-p_j)\right)^{T}}{\left|(p_l-p_j)\times(p_m-p_j)\right|}$$

$$\frac{\partial \tilde{p}_i}{\partial \boldsymbol{t}}=\boldsymbol{I}$$

$$\begin{aligned}\frac{\partial \tilde{p}_i}{\partial \delta\boldsymbol{\theta}}&=\frac{\partial\left(\exp(\delta\boldsymbol{\theta})\boldsymbol{R}p_i+\boldsymbol{t}\right)}{\partial \delta\boldsymbol{\theta}}\\&=\frac{\partial\left(\left(\boldsymbol{I}+\delta\boldsymbol{\theta}^{\wedge}\right)\boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}}\\&=\frac{\partial\left(\delta\boldsymbol{\theta}^{\wedge}\boldsymbol{R}p_i\right)}{\partial \delta\boldsymbol{\theta}}\\&=\frac{-\left(\boldsymbol{R}p_i\right)_{\times}\delta\boldsymbol{\theta}}{\partial \delta\boldsymbol{\theta}}\\&=-\left(\boldsymbol{R}p_i\right)_{\times}\end{aligned}$$

# 公式推导



- 面特征

最后得到

$$\frac{\partial d_{\mathcal{H}}}{\partial \boldsymbol{t}} = \frac{\boldsymbol{d}_{\mathcal{H}}^T}{d_{\mathcal{H}}} \frac{\left((p_l - p_j) \times (p_m - p_j)\right)^T}{|(p_l - p_j) \times (p_m - p_j)|}$$

$$\frac{\partial d_{\mathcal{H}}}{\partial \delta \boldsymbol{\theta}} = \frac{\boldsymbol{d}_{\mathcal{H}}^T}{d_{\mathcal{H}}} \frac{\left((p_l - p_j) \times (p_m - p_j)\right)^T}{|(p_l - p_j) \times (p_m - p_j)|} (-\boldsymbol{R} p_i)_{\times}$$

# 代码实现

## ●作业要求

1) 将激光里程计(aloam_laser_odometry_node)部分CERES优化问题的因子改为解析式求导。
2) 可以基于SE3扰动模型，此时优化参数块只有位姿，需要自定义类，继承自ceres::LocalParameterization，GlobalSize为7，LocalSize为6。
3) 也可以基于SO3扰动模型，此时优化参数块分为平移和旋转，旋转需要自定义类，继承自ceres::LocalParameterization，GlobalSize为4，LocalSize为3。
4) 框架里是分为两个参数块实现的，优化问题中需要分别添加，如下所示

```
ceres::Problem problem(problem_options);
problem.AddParameterBlock(para_q, 4, q_parameterization);
problem.AddParameterBlock(para_t, 3);
```

5) 框架中残差的定义可放在aloam_factor.hpp中，通过以下方式加入到优化问题中。

```
problem.AddResidualBlock(cost_function, loss_function, para_q, para_t);
```

6) 另外注意去畸变参数默认为s=1即可。

# 代码实现

## ●关于CERES

1) 图优化中的顶点是被优化的变量，CERES中被定义为ParameterBlock(参数块)。
2) 图优化中的边是观测对变量的约束，CERES中被定义为ResidualBlock(残差块)。
3) 旋转或者位姿参数块的自定义方法可参考Floam中的PoseSE3Parameterization类，主要实现Plus和ComputeJacobian两个函数。
4) 残差块的自定义方法可参考Floam中的EdgeAnalyticCostFunction，主要实现Evaluate函数。
5) 残差块关于LocalParameterization参数块的雅可比为dr/dx_local，在CERES中分为两部分dr/dx_global * dx_global/dx_local，其中dr/dx_global定义在残差块Evaluate函数中，dx_global/dx_local定义在参数块ComputeJacobian函数中。

# 代码实现

深蓝学院
shenlanxueyuan.com

## ●自定义旋转参数块

主要函数供参考

```cpp
bool PoseSO3Parameterization::Plus(const double *x, const double *delta, double *x_plus_delta) const
{
    Eigen::Quaterniond delta_q;
    getTransformFromSo3(Eigen::Map<const Eigen::Matrix<double,3,1>>(delta), delta_q);
    Eigen::Map<const Eigen::Quaterniond> quater(x);
    Eigen::Map<Eigen::Quaterniond> quater_plus(x_plus_delta);

    quater_plus = delta_q * quater;

    return true;
}


bool PoseSO3Parameterization::ComputeJacobian(const double *x, double *jacobian) const
{
    Eigen::Map<Eigen::Matrix<double, 4, 3, Eigen::RowMajor>> j(jacobian);
    (j.topRows(3)).setIdentity();
    (j.bottomRows(1)).setZero();

    return true;
}
```

# 代码实现

- 自定义线特征残差块

```cpp
virtual bool Evaluate(double const *const *parameters, double *residuals, double **jacobians) const
{
    Eigen::Map<const Eigen::Quaterniond> q_last_curr(parameters[0]);
    Eigen::Map<const Eigen::Vector3d> t_last_curr(parameters[1]);
    Eigen::Vector3d pi = q_last_curr * curr_point + t_last_curr; //new point
    Eigen::Vector3d nu = (pi - last_point_b).cross(pi - last_point_a);
    Eigen::Vector3d de = last_point_a - last_point_b;
    double nu_norm = nu.norm();
    double de_norm = de.norm();
    residuals[0] = nu_norm / de_norm;
```

```cpp
Eigen::Matrix3d skew_de = skew(de);
Eigen::Vector3d rp = q_last_curr * curr_point;
Eigen::Matrix3d skew_rp = skew(rp);

Eigen::Map<Eigen::Matrix<double, 1, 4, Eigen::RowMajor>> J_so3(jacobians[0]);
J_so3.setZero();
J_so3.block<1, 3>(0, 0) = nu.transpose() * skew_de * (-skew_rp) / (nu_norm * de_norm);

Eigen::Map<Eigen::Matrix<double, 1, 3, Eigen::RowMajor>> J_t(jacobians[1]);
J_t = nu.transpose() * skew_de / (nu_norm * de_norm);
```

# 代码实现

深蓝学院 shenlanxueyuan.com

- 自定义面特征残差块

```cpp
Eigen::Map<const Eigen::Quaterniond> q_last_curr(parameters[0]);
Eigen::Map<const Eigen::Vector3d> t_last_curr(parameters[1]);
Eigen::Vector3d pi = q_last_curr * curr_point + t_last_curr; //new point
double phi1 = (pi - last_point_j).dot(ljm_norm);

residuals[0] = std::fabs(phi1);
```

```cpp
if (residuals[0] != 0)
{
    phi1 = phi1 / residuals[0];
}
Eigen::Vector3d rp = q_last_curr * curr_point;
Eigen::Matrix3d skew_rp = skew(rp);

Eigen::Map<Eigen::Matrix<double, 1, 4, Eigen::RowMajor>> J_so3(jacobians[0]);
J_so3.setZero();
J_so3.block<1, 3>(0, 0) = phi1 * ljm_norm.transpose() * (-skew_rp);

Eigen::Map<Eigen::Matrix<double, 1, 3, Eigen::RowMajor>> J_t(jacobians[1]);
J_t = phi1 * ljm_norm.transpose();
```

# 在线问答

深蓝学院
shenlanxueyuan.com

# Q&A

# 感谢各位聆听！

Thanks for Listening