从计算机视觉(slam)和摄影测量两个维度进行 ba 算法原理推导

摄影测量作为历史悠久的学科，在 3D 视觉里面很多算法发挥着重要的作用；而 slam 的出现对摄影测量是某种程度上的冲击，但是并不能代表 slam 领域将会完全取代摄影测量领域，两者应该相互借鉴。以 bundle adjustment 为例出发点都是重投影误差，但是 slam 雅可比的计算是利用李群，而摄影测量的雅可比的计算是利用共线条件方程(非线性方程）泰勒展开，两者的最终结果相同，但是原理推导上有所差异：

令重投影误差: $r(\boldsymbol{\xi})=\mathbf{p}'-\mathbf{p}$,其中 $p^{'}$ 是像点近似值,p 是观测值, $\boldsymbol{\xi}=\left[\begin{array}{c}\boldsymbol{\rho}\\ \boldsymbol{\phi}\end{array}\right]\in\mathbb{R}^{6}$

本文从李群和共线方程两个学科角度来解释 bundle adjustment,并介绍 bundle adjustment 后如何精度评定及其 pose 增量变化在图像上的几何意义，而更多关于 bundle adjustment 的代码问题，如：

(1) pose 作为 const，只优化 3D points 和相机内参和畸变系数.

(2) 经典的 BA，pose 和 3Dpoints 以及相机内参和畸变系数都优化

(3) pose 部分参数优化，如只优化 rotation，固定 translation

(4) bundle adjustment 加上 GPS 约束

(5) bundle adjustment 加上 marker 约束

(6) bundle adjustment 在没有 gps、没有 gcp 的 case 加上真实世界中实际物体(如桌子长度)的 scale 约束

以上等等可以关注本人“视觉三维重建的关键技术与实现–colmap 代码解析”课程视频，具体课程介绍可以扫以下二维码：

1. 李群–利用矩阵

先看 ceres 中的定义如下：

```C++
local_jacobian = global_matrix * local_jacobian //ceres solver  local_parameterization.h
```

根据链式求导法则 $\frac{\partial r(\boldsymbol{\xi})}{\partial \xi}=\frac{\partial r(\boldsymbol{\xi})}{\partial \mathbf{P}}*\frac{\partial \bar{P}}{\partial \xi}$

(1) Jacobian 0:

$\mathbf{J}_0=\frac{\partial r(\boldsymbol{\xi})}{\partial \mathbf{p}'}=\mathbf{I}\in\mathbb{R}^{2\times 2}$

(2) Jacobian 1:

$\mathbf{J}_1=\frac{\partial \mathbf{p}'}{\partial \tilde{\mathbf{p}}'}=\frac{\partial(u,v)}{k*\partial\left(\bar{X}',\bar{Y}'\right)}$

其中 $\tilde{\mathbf{p}}'$ =K* $\tilde{\mathbf{p}}$,假设相机模型是 pinhole 模型,即 $\mathbf{K}=\left[\begin{array}{ccc}f_x & 0 & c_x\\ 0 & f_y & c_y\\ 0 & 0 & 1\end{array}\right]$

$\tilde{\mathbf{p}}=\left[\begin{array}{cc}f_x & 0\\ 0 & f_y\end{array}\right]*\left[\begin{array}{c}\bar{X}'\\ \bar{Y}'\end{array}\right]+\left[\begin{array}{c}\boldsymbol{c_x}\\ \boldsymbol{c_y}\end{array}\right]$

则 $\mathbf{J}_1=\left[\begin{array}{cc}f_x & 0\\ 0 & f_y\end{array}\right]\in\mathbb{R}^{2\times 2}$

(3) Jacobian 2:

由(2)继续进行链式求导:

$\mathbf{J}_2=\frac{\partial \bar{p}}{\partial \bar{p}}=\frac{\partial\left(\bar{X}^{\prime}, \bar{Y}^{\prime}\right)}{\partial(\bar{X}, \bar{Y}, \bar{Z})}$

其中 $\frac{\bar{X}}{\bar{Z}}=\bar{X}^{\prime} \quad \frac{\bar{Y}}{\bar{Z}}=\bar{Y}^{\prime}, \quad \bar{P}=[\bar{X}, \bar{Y}, \bar{Z}]^{\top}$

则上式为 $\left[\begin{array}{ccc}\frac{1}{\bar{Z}} & 0 & -\frac{\bar{X}}{\bar{Z}^2} \\ 0 & \frac{1}{\bar{Z}} & -\frac{\bar{Y}}{\bar{Z}^2}\end{array}\right] \in \mathbb{R}^{2 \times 3}$

(4) Jacobian3:

$$
\begin{aligned}
\mathbf{J}_3 & =\frac{\partial \overline{\mathbf{P}}}{\partial \boldsymbol{\xi}} \\
& =\frac{\partial(\mathbf{T} \cdot \mathbf{P})}{\partial \boldsymbol{\xi}} \\
& =\frac{\partial\left(\exp \left(\boldsymbol{\xi}^{\wedge}\right) \mathbf{P}\right)}{\partial \boldsymbol{\xi}}
\end{aligned}
$$

上式便是**经典的左扰动模型**,根据导数的思想 $\frac{d f}{d x}=\lim _{\Delta x \rightarrow 0} \frac{f(x+\Delta x)-f(x)}{\Delta x}$,将上式化为:

$$
\begin{aligned}
J_3=\frac{\partial \bar{P}}{\xi} & =\lim _{\Delta \xi->0} \frac{\exp \left(\xi^{\wedge}+\Delta \xi^{\wedge}\right) P-\exp \left(\xi^{\wedge}\right) P}{\Delta \xi} \\
& \approx \lim _{\Delta \xi->0} \frac{\left(I+\Delta \xi^{\wedge}\right) \exp \left(\xi^{\wedge}\right) P-\exp \left(\xi^{\wedge}\right) P}{\Delta \xi} \\
& =\lim _{\Delta \xi->0} \frac{\Delta \xi^{\wedge} \exp \left(\xi^{\wedge}\right) P}{\Delta \xi}
\end{aligned}
$$

继续化简得到如下:

$$
J_3=\lim _{\Delta \xi \rightarrow 0} \frac{\left[\begin{array}{cc}\Delta \phi^{\wedge} & \Delta \rho \\ 0^T & 0\end{array}\right]\left[\begin{array}{c}R P+t \\ 1\end{array}\right]}{\Delta \xi}
$$

$$
=\lim _{\Delta \xi->0} \frac{\left[\begin{array}{c}\Delta \phi^{\wedge}(R P+t)+\Delta \rho \\ 0\end{array}\right]}{\Delta \xi}
$$

令 $R P+t=\left[\begin{array}{c}\bar{X} \\ \bar{Y} \\ \bar{Z}\end{array}\right]=\bar{P}$

则 $\Delta \phi^{\wedge}(R P+t)+\Delta \rho=\left[\begin{array}{c}-\Delta \phi_3 \bar{Y}+\Delta \phi_2 \bar{Z}+\Delta \rho_1 \\ \Delta \phi_3 \bar{X}-\Delta \phi_1 \bar{Z}+\Delta \rho_2 \\ -\Delta \phi_2 \bar{X}+\Delta \phi_1 \bar{Y}+\Delta \rho_3\end{array}\right]$

上式对 $\Delta \xi$ (6*1)求导可得:

i.对 $\Delta \rho_{1-3}$ 分别求导得 $\left[\begin{array}{ccc}1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1\end{array}\right]$

ii. 对 $\Delta \phi_{1-3}$ 分别求导可得 $\left[\begin{array}{ccc}0 & \bar{Z} & -\bar{Y} \\ -\bar{Z} & 0 & \bar{X} \\ \bar{Y} & -\bar{X} & 0\end{array}\right]=-\bar{P}^{\wedge}$

故 $\mathbf{J}_3=\left[\begin{array}{ll}\mathbf{I} & -\bar{P}^{\wedge}\end{array}\right]$

$$
=\left[\begin{array}{cccccc}1 & 0 & 0 & 0 & \bar{Z} & -\bar{Y} \\ 0 & 1 & 0 & -\bar{Z} & 0 & \bar{X} \\ 0 & 0 & 1 & \bar{Y} & -\bar{X} & 0\end{array}\right] \in \mathbb{R}^{3 \times 6}
$$

故最终的雅可比

$\mathbf{J}(\boldsymbol{\xi})=\mathbf{J}_0 \cdot \mathbf{J}_1 \cdot \mathbf{J}_2 \cdot \mathbf{J}_3$

2.摄影测量–利用共线条件方程

$x-x_0=f \frac{a_1\left(X-X_s\right)+b_1\left(Y-Y_s\right)+c_1\left(Z-Z_s\right)}{a_3\left(X-X_s\right)+b_3\left(Y-Y_s\right)+c_3\left(Z-Z_s\right)}=f \frac{\bar{X}}{\bar{Z}}$

$y-y_0=f \frac{a_2\left(X-X_s\right)+b_2\left(Y-Y_s\right)+c_2\left(Z-Z_s\right)}{a_2\left(X-X_n\right)+b_2\left(Y-Y_n\right)+c_2\left(Z-Z_n\right)}=f \frac{\bar{Y}}{\bar{Z}}$ (1)

上式子就是经典的共线条件方程

其中

$$\begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \end{bmatrix} = \begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ a_3 & b_3 & c_3 \end{bmatrix} \begin{bmatrix} X - X_s \\ Y - Y_s \\ Z - Z_s \end{bmatrix} = \boldsymbol{R} \begin{bmatrix} X - X_s \\ Y - Y_s \\ Z - Z \end{bmatrix}$$

这里令摄影测量的转角系统和 CV 一致,即 R–P–Y 顺序,即:

$$R = R(r,p,y) = R_3(y)R_2(p)R_1(r) = \begin{bmatrix} \boldsymbol{r}_1^\top \\ \boldsymbol{r}_2^\top \\ \boldsymbol{r}_3^\top \end{bmatrix}$$

平差模型：观测值 + 观测值改正数=近似值 + 近似值改正数

$$x + v_x = (x) + dx$$

$$y + v_y = (y) + dy$$

其中令角元素都是小角

$$\begin{aligned} \frac{\partial x}{\partial X_s} &= \frac{f}{\bar{Z}^2}\left(\frac{\partial \bar{X}}{\partial X_s}\bar{Z} - \frac{\partial \bar{Z}}{\partial X_s}\bar{X}\right) \\ &= -\frac{f}{\bar{Z}^2}\left(-a_1\bar{Z} + a_3\bar{X}\right) \\ &= \frac{1}{\bar{Z}}\left(a_1 f + f\frac{\bar{X}}{\bar{Z}}a_3\right) \\ &= \frac{1}{\bar{Z}}\left[-a_1 f + a_3\left(x - x_0\right)\right] = -\frac{f}{\bar{Z}} \end{aligned}$$

也是利用链式求导原则，以下分别对 roll,pitch 和 yaw 求导(对应 无人机 ω (旁向倾角)，ϕ (航向倾角)，κ (像片旋角))

$$\frac{\partial x}{\partial r} = \frac{f}{\bar{Z}^2}\left(\frac{\partial \bar{X}}{\partial r}\bar{Z} - \frac{\partial \bar{Z}}{\partial r}\bar{X}\right) = f\frac{\bar{X}\bar{Y}}{\bar{Z}^2}$$

$$\frac{\partial \begin{bmatrix} \bar{X} \\ \bar{Y} \\ \bar{Z} \end{bmatrix}}{\partial r} = \boldsymbol{R}_y R_p \boldsymbol{R}_r R_r^{-1}\frac{\partial R_r}{\partial r}\begin{bmatrix} X - X_s \\ Y - Y_s \\ Z - Z_s \end{bmatrix} = \boldsymbol{R}\begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}\begin{bmatrix} X - X_s \\ Y - Y_s \\ Z - Z_s \end{bmatrix}$$

以此类推,最终得到误差方程的系数即雅可比矩阵为:

$$\begin{array}{cc} & \begin{array}{cccccc} \Delta X_O & \Delta Y_O & \Delta Z_O & \Delta roll & \Delta pitch & \Delta yaw \end{array} \\ B = & \begin{bmatrix} -\frac{f}{Z_i - Z_O} & 0 & -\frac{x_i'}{Z_i - Z_O} & \frac{x_i' y_i'}{f} & -f\left(1 + \frac{x_i^2}{f^2}\right) & y_i' \\ 0 & -\frac{f}{Z_i - Z_O} & -\frac{y_i'}{Z_i - Z_O} & f\left(1 + \frac{y_i^2}{f^2}\right) & -\frac{x_i' y_i'}{f} & -x_i' \end{bmatrix} \end{array}$$

slam 误差方程系数:

$$-\begin{bmatrix} f_x\frac{1}{Z'} & 0 & -f_x\frac{X'}{Z'^2} & -f_x\frac{X'Y'}{Z'^2} & f_x\left(1 + \frac{X'^2}{Z'^2}\right) & -f_x\frac{Y'}{Z'} \\ 0 & f_y\frac{1}{Z'} & -f_y\frac{Y'}{Z'^2} & -f_y\left(1 + \frac{Y'^2}{Z'^2}\right) & f_y\frac{X'Y'}{Z'^2} & f_y\frac{X'}{Z'} \end{bmatrix}$$ (其中负号代表观测值减去近似值)

对比上式 B 矩阵,可见 photogrammetry 和 slam 的雅可比(误差方程系数)完全一样!

3. 精度评定–协方差计算

常见的 BA 代码中,通常认为观测值是等权的,即 $\sigma_0^2 = \sigma_{x_i'}^2 = \sigma_{y_i'}^2$ => $p_i = \sigma_0^2/\sigma_{x_i'}^2 = \sigma_0^2/\sigma_{y_i'}^2 = 1$.

而 6dof 的权的求解是从误差方程出发:

(1) 残差方程为 $\widehat{\boldsymbol{v}}_i = \boldsymbol{f}_i(\widehat{\boldsymbol{x}}) - \boldsymbol{l}_i$

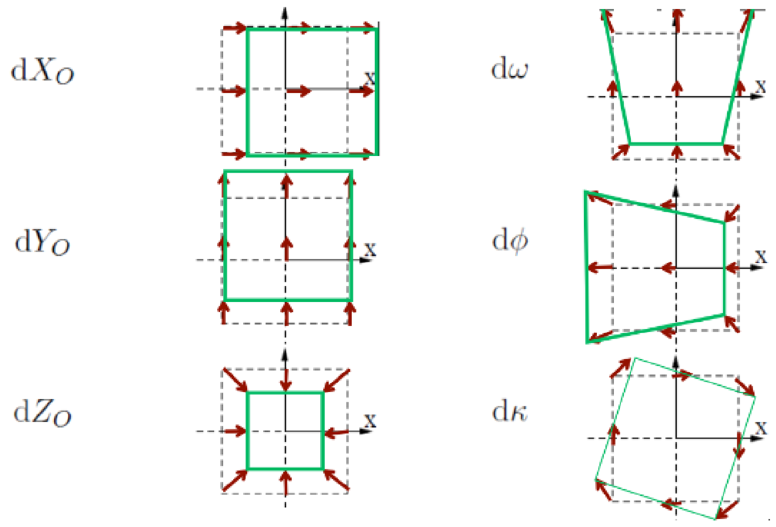
(2) 法方程系数构建 $\Omega = \sum_i \widehat{\boldsymbol{v}}_i^\top P_i \widehat{\boldsymbol{v}}_i$,这里的 P_i 是观测值的权

(3) 计算单位权中误差 $\widehat{\sigma}_0^2 = \frac{\Omega}{r} = \frac{\Omega}{2I-6}$,其中 I 代表像点个数

(4) 计算协方差 $\widehat{\Sigma}_{\widehat{x}\widehat{x}} = \widehat{\sigma}_0^2 N^{-1}$,这里的 N 为法方程系数的逆

(5) 6dof 的方差取 $\widehat{\Sigma}_{\widehat{x}\widehat{x}}$ 的对角线即可

4. BA 6dof 的增量在图像上呈现的几何意义



5. BA 示例代码（只优化 R，t）

```
// 经典的ba 代码采用摄影测量方式编写如下：
#include <iostream>
#include <opencv2/core/core.hpp>
#include <ceres/ceres.h>
#include <chrono>
#include<eigen3/Eigen/Core>
#include<eigen3/Eigen/Dense>
#include<fstream>
#include<math.h>
using namespace std;
double sumVector(vector<double> x)
{
     double sum=0.0;
     for(int i=0; i<x.size();++i)
     {
         sum+=x[i];
     }
     return sum/x.size();
}
// 代价函数的计算模型
struct Resection
{
    Resection ( double X,double Y,double Z,double x, double y,double f ) :_X(X),_Y(Y),_Z(Z), _x ( x ), _y ( y ),_f(f) {}
    // 残差的计算
    template <typename T>
    bool operator() (const T* const camPose, T* residual ) const     // 残差
    {
            T Xs=camPose[0];
            T Ys=camPose[1];
            T Zs=camPose[2];
            T w=camPose[3];
            T p=camPose[4];
            T k=camPose[5];
            T a1=cos(k)*cos(p);
            T b1=-sin(k)*cos(w) + sin(p)*sin(w)*cos(k);
            T c1=sin(k)*sin(w) + sin(p)*cos(k)*cos(w);
            T a2=sin(k)*cos(p);
            T b2=sin(k)*sin(p)*sin(w) + cos(k)*cos(w);
            T c2=sin(k)*sin(p)*cos(w) - sin(w)*cos(k);
            T a3=-sin(p);
            T b3=sin(w)*cos(p);
            T c3=cos(p)*cos(w);
//            R=rotationVectorToMatrix(omega,pho,kappa);
            residual[0]=T(_x)-T(_f)*T((a1*(_X-Xs)+b1*(_Y-Ys)+c1*(_Z-Zs))/(a3*(_X-Xs)+b3*(_Y-Ys)+c3*(_Z-Zs)));
            residual[1]=T(_y)-T(_f)*T((a2*(_X-Xs)+b2*(_Y-Ys)+c2*(_Z-Zs))/(a3*(_X-Xs)+b3*(_Y-Ys)+c3*(_Z-Zs)));
            return true;
    }
private:
    const double _x;
    const double _y;
    const double _f;
    const double _X;
    const double _Y;
    const double _Z;
};

int main ( int argc, char** argv )
{
    google::InitGoogleLogging(argv[0]);
    //read file
    string filename=argv[1];
    ifstream fin(filename.c_str());
    string line;
    vector<double> x;
    vector<double> y;
    vector<double> X;
    vector<double> Y;
    vector<double> Z;
    while(getline(fin,line))
    {
        char* pEnd;
        double a,b,c,d,e;
```

```
        a=strtod(line.c_str(),&pEnd);
        b=strtod(pEnd,&pEnd);
        c=strtod(pEnd,&pEnd);
        d=strtod(pEnd,&pEnd);
        e=strtod(pEnd,nullptr);
        x.push_back(a);
        y.push_back(b);
        X.push_back(c);
        Y.push_back(d);
        Z.push_back(e);
    }
    //初始化参数
    double camPose[6]={0};
    camPose[0]=sumVector(X);
    camPose[1]=sumVector(Y);
    camPose[2]=sumVector(Z);
    double f = 153.24; //mm
//    camPose[2]=50*f;
    //构建最小二乘
    ceres::Problem problem;
    try
    {
        for(int i=0;i<x.size();++i)
        {
            ceres::CostFunction *costfunction=new ceres::AutoDiffCostFunction<Resection,2,6>(new Resection(X[i],Y[i],Z[i
            //将残差方程和观测值加入到problem,nullptr表示核函数为无，
            problem.AddResidualBlock(costfunction,nullptr,camPose);
        }
    }
    catch(...)
    {
        cout<<"costFunction error"<<endl;
    }

    // 配置求解器
    ceres::Solver::Options options;     // 这里有很多配置项可以填
    options.linear_solver_type = ceres::DENSE_QR;  // 增量方程如何求解
    options.minimizer_progress_to_stdout = true;   // 输出到cout
//    options.max_num_iterations=25;
    ceres::Solver::Summary summary;                // 优化信息
    chrono::steady_clock::time_point t1 = chrono::steady_clock::now();
    ceres::Solve ( options, &problem, &summary );  // 开始优化
    chrono::steady_clock::time_point t2 = chrono::steady_clock::now();
    chrono::duration<double> time_used = chrono::duration_cast<chrono::duration<double>>( t2-t1 );
    cout<<"solve time cost = "<<time_used.count()<<" seconds. "<<endl;

    // 输出结果
    cout<<summary.BriefReport() <<endl;
    cout<<"*************结果****************"<<endl;
    cout<<"estimated Xs,Ys,Zs,omega,pho,kappa = ";
    for ( auto p:camPose) cout<<p<<" ";
    cout<<endl;

    return 0;
}

//输入数据格式如下：
-86.15 -68.99 36589.41 25273.32 2195.17
-53.40 82.21 37631.08 31324.51 728.69
-14.78 -76.63 39100.97 24934.98 2386.50
10.46 64.43 40426.54 30319.81 757.31
```