Jindong_Shi
2021-11-05

第七节课习题

2. Bundle Adjustment

2.1 文献阅读

1) 为何说 Bundle Adjustment is slow 是不对的？

One aim of this paper is to correct a number of misconceptions that seem to be common in the vision literature:

- ***"Optimization / bundle adjustment is slow"*:** Such statements often appear in papers introducing yet another heuristic Structure from Motion (SFM) iteration. The claimed slowness is almost always due to the unthinking use of a general-purpose optimization routine that completely ignores the problem structure and sparseness. Real bundle routines are *much* more efficient than this, and usually considerably more efficient and flexible than the newly suggested method (§6, 7). That is why bundle adjustment remains the dominant structure refinement technique for real applications, after 40 years of research.

根据文献[1]的说法，之前研究者们并未关注到增量方程矩阵 H 自身的稀疏性结构，误认为解决此优化问题是十分消耗计算量的。事实上，根据 Schur trick 等利用稀疏性加速的方法，可以让 BA 问题的实时求解成为可能。

2) BA 中有哪些需要注意参数化的地方？Pose 和 Point 各有哪些参数化方式？有何优缺点。

The bundle adjustment parameter space is generally a high-dimensional nonlinear manifold — a large Cartesian product of projective 3D feature, 3D rotation, and camera calibration manifolds, perhaps with nonlinear constraints, *etc*. The state x is not strictly speaking a vector, but rather a point in this space. Depending on how the entities that it contains are represented, x can be subject to various types of complications including singularities, internal constraints, and unwanted internal degrees of freedom. These arise because geometric entities like rotations, 3D lines and even projective points and planes, do not have simple global parametrizations. Their local parametrizations are nonlinear, with singularities that prevent them from covering the whole parameter space uniformly (*e.g.* the many variants on Euler angles for rotations, the singularity of affine point coordinates at infinity). And their global parametrizations either have constraints (*e.g.* quaternions with $\|\boldsymbol{q}\|^2 = 1$), or unwanted internal degrees of freedom (*e.g.* homogeneous projective quantities have a scale factor freedom, two points defining a line can slide along the line). For more complicated compound entities such as matching tensors and assemblies of 3D features linked by coincidence, parallelism or orthogonality constraints, parametrization becomes even more delicate.

参数化可能很麻烦.

BA 当中待优化的参数的状态空间 (包括特征点位置，相机内外参数) 往往位于一个非线性流形之上，它们在优化时会出现奇异性，受约束和自由度缺失等问题。比如单位四元数模长为 1，直线上的特征点存在额外的缩放自由度，旋转矩阵和变换矩阵只存在李群流形之上，无穷远处的点存在奇异性等等问题。另一方面当这些优化参数之间存在过多上面这些相关性和奇异性时，会造成求解的增量矩阵陷入病态，导致无法求解。

3D points: Even for calibrated cameras, vision geometry and visual reconstructions are intrinsically projective. If a 3D $(X\ Y\ Z)^\top$ parametrization (or equivalently a homogeneous affine $(X\ Y\ Z\ 1)^\top$ one) is used for very distant 3D points, large X, Y, Z displacements are needed to change the image significantly. *I.e.*, in $(X\ Y\ Z)$ space the cost function becomes very flat and steps needed for cost adjustment become very large for distant points. In comparison, with a homogeneous projective parametrization $(X\ Y\ Z\ W)^\top$, the behaviour near infinity is natural, finite and well-conditioned so long as the normalization keeps the homogeneous 4-vector finite at infinity (by sending $W \to 0$ there). In fact, there is no immediate visual distinction between the images of real points near infinity and virtual ones 'beyond' it (all camera geometries admit such virtual points as *bona fide* projective constructs). The optimal reconstruction of a real 3D point may even be virtual in this sense, if image noise happens to push it 'across infinity'. Also, there is nothing to stop a reconstructed point wandering beyond infinity and back during the optimization. This sounds bizarre at first, but it is an inescapable consequence of the fact that the natural geometry and error model for visual reconstruction is projective rather than affine. Projectively, *infinity is just like any other place*. Affine parametrization $(X\ Y\ Z\ 1)^\top$ is acceptable for points near the origin with close-range convergent camera geometries, but it is disastrous for distant ones because it artificially cuts away half of the natural parameter space, and hides the fact by sending the resulting edge to infinite parameter values. Instead, you should use a homogeneous parametrization $(X\ Y\ Z\ W)^\top$ for distant points, *e.g.* with spherical normalization $\sum X_i^2 = 1$.

对于 3D points 的参数化，我们可以既可以使用齐次放射(homogeneous affine)形式(X Y Z 1), 也可以使用齐次投影(homogeneous projective)的形式(X Y Z W)。前者的灾难性缺点是无法处理无穷远处或非常远的点，后者可以通过将 W 变换成 0 解决这个问题，无穷远处点在这种形式的表示下同其他任何地方的点是一样可靠的。

Rotations: Similarly, experience suggests that quasi-global 3 parameter rotation parametrizations such as Euler angles cause numerical problems unless one can be certain to avoid their singularities and regions of uneven coverage. Rotations should be parametrized using either quaternions subject to $\|\boldsymbol{q}\|^2 = 1$, or local perturbations $\boldsymbol{R}\,\delta\boldsymbol{R}$ or $\delta\boldsymbol{R}\,\boldsymbol{R}$ of an existing rotation R, where $\delta\boldsymbol{R}$ can be any well-behaved 3 parameter small rotation approximation, *e.g.* $\delta\boldsymbol{R} = (\boldsymbol{I} + [\,\delta\boldsymbol{r}\,]_\times)$, the Rodriguez formula, local Euler angles, *etc*.

对于 Pose 而言，3 参数形式(比如欧拉角)存在奇异性问题，所以我们更倾向于使用四元数表示旋转，或通过李代数的形式来表示旋转。

Jindong_Shi
2021-11-05

3) * 本文写于 2000 年，但是文中提到的很多内容在后面十几年的研究中得到了印证。你能看到哪些方向在后续工作中有所体现？请举例说明。

model might be a collection of isolated 3D features, *e.g.*, points, lines, planes, curves, or surface patches. However, far more complicated scene models are possible, involving, *e.g.*, complex objects linked by constraints or articulations, photometry as well as geometry, dynamics, *etc*. One of the great strengths of adjustment computations — and one reason for thinking that they have a considerable future in vision — is their ability to take such complex and heterogeneous models in their stride. Almost any *predictive parametric* model can be handled, *i.e.* any model that *predicts* the values of some known measurements or descriptors on the basis of some continuous *parametric* representation of the world, which is to be estimated from the measurements.

Similarly, many possible camera models exist. Perspective projection is the standard, but the affine and orthographic projections are sometimes useful for distant cameras, and more exotic models such as push-broom and rational polynomial cameras are needed for certain applications [56, 63]. In addition to pose (position and orientation), and simple internal parameters such as focal length and principal point, real cameras also require various types of **additional parameters** to model internal aberrations such as radial distortion [17, 18, 19, 100, 69, 5].

BA 的方法由于其灵活性，精确性，实时性(非必要，取决于应用场合) 如今被广泛应用于三维重建当中。

tions. The abstract structure of the measurement network can be characterized graphically by the **network graph** (top left), which shows which features are seen in which images, and the **parameter connection graph** (top right) which details the sparse structure by showing which parameter blocks have direct interactions. Blocks are linked if and only if they jointly influence at least one observation. The cost function Jacobian (bottom left) and Hessian (bottom right) reflect this sparse structure. The shaded boxes correspond to non-zero blocks of matrix entries. Each block of rows in the Jacobian corresponds to an observed image feature and contains contributions from each of the parameter blocks that influenced this observation. The Hessian contains an off-diagonal block for each edge of the parameter connection graph, *i.e.* for each pair of parameters that couple to at least one common feature / appear in at least one common cost contribution[10].

Bundle problems are by no means limited to the above structures. For example, for more complex scene models with moving or articulated objects, there will be additional connections to object pose or joint angle nodes, with linkages reflecting the kinematic chain structure of the scene. It is often also necessary to add constraints to the adjustment, *e.g.* coplanarity of certain points. One of the greatest advantages of the bundle technique is its ability to adapt to almost arbitrarily complex scene, observation and constraint models.

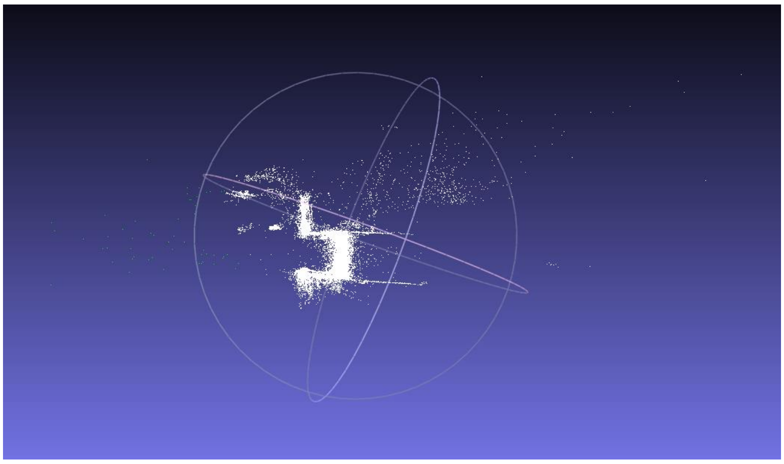
BA 问题对应了一个十分明显的图模型结构。我们对 BA 问题的优化等价于对图模型的优化，对应了后来图优化的理论。

2.2 BAL-dataset

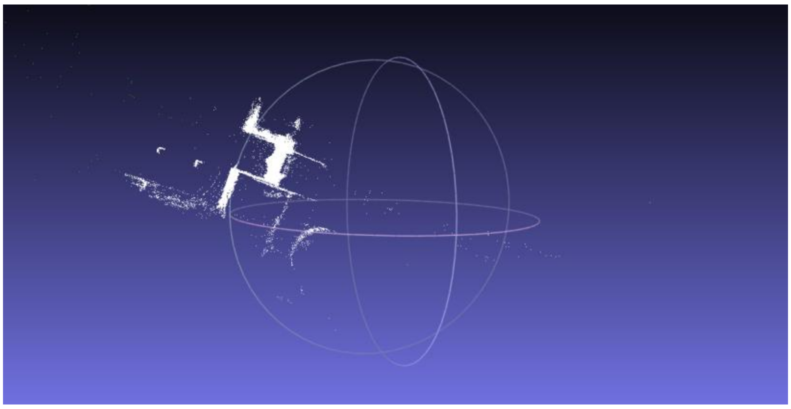
源文件 BA_in_large.cpp 位于文件夹 BA_in_large 内，可执行文件位于文件夹 BA_in_large/OUTPUT 内。计算结果如下:

```
jindong@jindong-virtual-machine:~/SLAM/Chap7/L7_code/BA_in_large/OUTPUT$ ./BA_in_large
pose number: 52
point number: 64053
observations number: 347173
iteration= 0     chi2= 33290614.470124   time= 1.60786   cumTime= 1.60786       edges= 347173    schur= 1        lambda= 207456.754910   levenbergIter= 1
iteration= 1     chi2= 11123023.659253   time= 1.14143   cumTime= 2.74929        edges= 347173   schur= 1         lambda= 69152.251637    levenbergIter= 1
iteration= 2     chi2= 5049836.940214    time= 1.77827   cumTime= 4.52756        edges= 347173   schur= 1        lambda= 23050.750546    levenbergIter= 1
iteration= 3     chi2= 2370357.197883    time= 1.36783   cumTime= 5.89539        edges= 347173   schur= 1        lambda= 7683.583515     levenbergIter= 1
iteration= 4     chi2= 1331546.029231    time= 1.16225   cumTime= 7.05764        edges= 347173   schur= 1        lambda= 2561.194505     levenbergIter= 1
iteration= 5     chi2= 864232.191903     time= 1.26846   cumTime= 8.3261         edges= 347173   schur= 1        lambda= 853.731502      levenbergIter= 1
iteration= 6     chi2= 662315.739410     time= 1.33565   cumTime= 9.66174        edges= 347173   schur= 1        lambda= 284.577167      levenbergIter= 1
iteration= 7     chi2= 570321.534443     time= 1.09118   cumTime= 10.7529        edges= 347173   schur= 1        lambda= 94.859056       levenbergIter= 1
iteration= 8     chi2= 514080.470480     time= 1.07163   cumTime= 11.8245        edges= 347173   schur= 1        lambda= 31.619685       levenbergIter= 1
iteration= 9     chi2= 477563.683883     time= 1.08572   cumTime= 12.9103        edges= 347173   schur= 1        lambda= 10.539895       levenbergIter= 1
iteration= 10    chi2= 447955.853829     time= 1.08149   cumTime= 13.9918        edges= 347173   schur= 1        lambda= 3.513298        levenbergIter= 1
iteration= 11    chi2= 406982.776336     time= 1.08993   cumTime= 15.0817        edges= 347173   schur= 1        lambda= 1.171099        levenbergIter= 1
iteration= 12    chi2= 345742.102298     time= 1.10244   cumTime= 16.1841        edges= 347173   schur= 1        lambda= 0.390366        levenbergIter= 1
iteration= 13    chi2= 284218.280087     time= 1.08497   cumTime= 17.2691        edges= 347173   schur= 1        lambda= 0.145228        levenbergIter= 1
iteration= 14    chi2= 252028.958527     time= 1.24081   cumTime= 18.5099        edges= 347173   schur= 1        lambda= 0.096819        levenbergIter= 1
iteration= 15    chi2= 209839.529217     time= 1.11248   cumTime= 19.6224        edges= 347173   schur= 1        lambda= 0.055855        levenbergIter= 1
iteration= 16    chi2= 199360.029672     time= 1.08891   cumTime= 20.7113        edges= 347173   schur= 1        lambda= 0.037237        levenbergIter= 1
iteration= 17    chi2= 194913.731662     time= 1.10018   cumTime= 21.8115        edges= 347173   schur= 1        lambda= 0.024824        levenbergIter= 1
iteration= 18    chi2= 192174.259748     time= 1.08162   cumTime= 22.8931        edges= 347173   schur= 1        lambda= 0.016550        levenbergIter= 1
iteration= 19    chi2= 189668.511927     time= 1.08788   cumTime= 23.981         edges= 347173   schur= 1        lambda= 0.011033        levenbergIter= 1
The Optimization of BA in Large has been finished!
jindong@jindong-virtual-machine:~/SLAM/Chap7/L7_code/BA_in_large/OUTPUT$
```

优化前:



优化后:



很明显，优化后的结果更好。

雅可比的推导过程如下:

设相机(camera)的待优化参数为 $y=[\rho,\phi,f,k_1,k_2]^T$，共9维，其中位姿 $\xi=[\phi\ \rho]$

设路标点(Point)的待优化参数为 $P_w=[X_w,Y_w,Z_w]^T$，共3维

记变换到相机坐标系下的路标点坐标为 P_c：

匹配G2O对se(3)的定义：旋转在前，平移在后！

$$P_c=\left(\exp(\xi^\wedge)P_w\right)_{1:3}=[X_c,Y_c,Z_c]^T$$

根据"Bundle Adjustment in the Large"数据集的定义：

先将 P_c 归一化，得 $P=-P_c/P_c.z_c=\left[-\frac{X_c}{Z_c},-\frac{Y_c}{Z_c},-1\right]^T_{1:2}=\left[-\frac{X_c}{Z_c},\frac{-Y_c}{Z_c}\right]^T$

将其投影到相机平面上，考虑畸变后的投影坐标 $[u_c,v_c]^T$ 为： $\Rightarrow \|P\|^2=\left(\frac{X_c}{Z_c}\right)^2+\left(\frac{Y_c}{Z_c}\right)^2$

$$u_c=f x\left[1+k_1\|P\|^2+k_2\|P\|^4\right]=-\frac{fX_c}{Z_c}\left[1+k_1\left[\left(\frac{X_c}{Z_c}\right)^2+\left(\frac{Y_c}{Z_c}\right)^2\right]+k_2\left[\left(\frac{X_c}{Z_c}\right)^2+\left(\frac{Y_c}{Z_c}\right)^2\right]^2\right]$$

同理

$$v_c=-\frac{fY_c}{Z_c}\left[1+k_1\left[\left(\frac{X_c}{Z_c}\right)^2+\left(\frac{Y_c}{Z_c}\right)^2\right]+k_2\left[\left(\frac{X_c}{Z_c}\right)^2+\left(\frac{Y_c}{Z_c}\right)^2\right]^2\right]$$

我们定义误差为观测值减去估计值：

$$e=m-\begin{bmatrix}u_c\\v_c\end{bmatrix}$$，其中 m 为观测到的投影坐标

① 我们先求误差 e 对相机待优化参数 y 的偏导，由于位姿 ξ 无法求偏导，所以我们先用李代数处理 e 对位姿 ξ 的偏导。我们对 $\xi^\wedge$ 左扰动 $\delta\xi$，利用链式法则，有

$$\frac{\partial e}{\partial\delta\xi}=\lim_{\delta\xi\to0}\frac{e(\delta\xi\oplus\xi)}{\delta\xi}=\frac{\partial e}{\partial P_c}\cdot\frac{\partial P_c}{\partial\xi}$$

首先：我们定义 P 的模长为 n： $n^2=\|P\|^2=x^2+y^2+z^2=\frac{X_c^2}{Z_c^2}+\frac{Y_c^2}{Z_c^2}+1=\frac{X_c^2+Y_c^2+Z_c^2}{Z_c^2}$

定义 $q=r(p)=1+k_1\|P\|^2+k_2\|P\|^4$

那么：

$$\frac{\partial e}{\partial P_c}=-\begin{bmatrix}\frac{\partial u_c}{\partial X_c}&\frac{\partial u_c}{\partial Y_c}&\frac{\partial u_c}{\partial Z_c}\\\frac{\partial v_c}{\partial X_c}&\frac{\partial v_c}{\partial Y_c}&\frac{\partial v_c}{\partial Z_c}\end{bmatrix}$$

$$\frac{\partial u_c}{\partial X_c}=-\frac{f}{Z_c}\cdot q-\frac{fX_c}{Z_c}\left[2k_1\frac{X_c}{Z_c}\cdot\frac{1}{Z_c}+2k_2n^2\cdot2\frac{X_c}{Z_c}\cdot\frac{1}{Z_c}\right]$$

$$=-\frac{f}{Z_c}\cdot q-\frac{fX_c}{Z_c}\cdot\frac{2k_1X_c+4k_2X_cn^2}{Z_c^2}$$

$$=-\frac{f}{Z_c}\cdot q-\frac{2fX_c^2(k_1+2k_2n^2)}{Z_c^3}$$

$$\frac{\partial u_c}{\partial Y_c}=-\frac{fX_c}{Z_c}\left[2k_1\frac{Y_c}{Z_c}\cdot\frac{1}{Z_c}+2k_2n^2\cdot2\frac{Y_c}{Z_c}\cdot\frac{1}{Z_c}\right]$$

$$=-\frac{fX_c}{Z_c}\cdot\frac{2Y_c(k_1+2k_2n^2)}{Z_c^2}$$

$$=-\frac{fX_c}{Z_c^3}\cdot2Y_c(k_1+2k_2n^2)$$

$$\frac{\partial u_c}{\partial Z_c}=\frac{fX_c}{Z_c^2}\cdot q-\frac{fX_c}{Z_c}\left[2k_1\frac{X_c}{Z_c}\cdot\frac{-X_c}{Z_c^2}+2k_1\frac{Y_c}{Z_c}\cdot\frac{-Y_c}{Z_c^2}+2k_2n^2\left(2\frac{X_c}{Z_c}\cdot\frac{-X_c}{Z_c^2}+2\frac{Y_c}{Z_c}\frac{-Y_c}{Z_c^2}\right)\right]$$

$$=\frac{fX_c}{Z_c^2}\cdot\left[q+2n^2(k_1+2k_2n^2)\right]$$

$$\frac{\partial v_c}{\partial x_c} = \frac{-fy_c}{z_c}\left[2k_1\frac{x_c}{z_c}\cdot\frac{1}{z_c} + k_2\cdot 2n^2\cdot 2\frac{x_c}{z_c}\cdot\frac{1}{z_c}\right]$$

$$= \frac{-fy_c}{z_c}\ \frac{2x_c(k_1+2k_2n^2)}{z_c^2}$$

$$= \frac{-fx_c}{z_c^3}\ y_c(k_1+2k_2n^2)$$

$$\frac{\partial v_c}{\partial y_c} = -\frac{f}{z_c}q - \frac{fy_c}{z_c}\left[2k_1\frac{y_c}{z_c}\cdot\frac{1}{z_c} + 2k_2n^2\cdot 2\frac{y_c}{z_c}\cdot\frac{1}{z_c}\right]$$

$$= \frac{-fq}{z_c} - \frac{2fy_c^2(k_1+2k_2n^2)}{z_c^3}$$

$$\frac{\partial v_c}{\partial z_c} = \frac{fy_c}{z_c^2}\cdot q - \frac{fy_c}{z_c}\left[2k_1\frac{x_c}{z_c}\frac{-x_c}{z_c^2} + 2k_1\frac{y_c}{z_c}\cdot\frac{-y_c}{z_c^2} + 2k_2n^2\left(2\frac{x_c}{z_c}\cdot\frac{-x_c}{z_c^2} + 2\frac{y_c}{z_c}\frac{-y_c}{z_c^2}\right)\right]$$

$$= \frac{fy_c}{z_c^2}\left[q + 2n^2(k_1+2k_2n^2)\right]$$

由于 $k_1+2k_2n^2$ 多次出现，所以我们设 $w = k_1+2k_2n^2$

然后再求 $\frac{\partial P_c}{\partial \delta \xi}$：

$$\frac{\partial P_c}{\partial \delta \xi} = \frac{\partial (TP_w)}{\partial \delta \xi} = (TP_w)^{\odot} = \begin{bmatrix} -P_c^{\wedge} & I \\ 0^T & 0^T \end{bmatrix}$$

取出前三维：

$$\frac{\partial P_c}{\partial \delta \xi} = [-P_c^{\wedge}, \quad I]$$

然后求 $\frac{\partial e}{\partial f} = \begin{bmatrix} -\frac{\partial u}{\partial f} \\ -\frac{\partial v}{\partial f} \end{bmatrix} = \begin{bmatrix} \frac{x}{z}[1 + k_1 r^2 + k_2 r^4] \\ \frac{y}{z}[1 + k_1 r^2 + k_2 r^4] \end{bmatrix}$

$$\frac{\partial e}{\partial k_1} = \begin{bmatrix} -\frac{\partial u}{\partial k_1} \\ -\frac{\partial v}{\partial k_1} \end{bmatrix} = \begin{bmatrix} \frac{f x r^2}{z} \\ \frac{f y r^2}{z} \end{bmatrix}$$

$$\frac{\partial e}{\partial k_2} = \begin{bmatrix} -\frac{\partial u}{\partial k_2} \\ -\frac{\partial v}{\partial k_2} \end{bmatrix} = \begin{bmatrix} \frac{f x r^4}{z} \\ \frac{f y r^4}{z} \end{bmatrix}$$

最终得到误差对相机 (camera) 所有待优化参数 y 的雅可比，其维度为 2×9

$$J_{ey} = \left[\frac{\partial e}{\partial P}\frac{\partial P}{\partial \delta \xi}, \quad \frac{\partial e}{\partial f}, \quad \frac{\partial e}{\partial k_1}, \quad \frac{\partial e}{\partial k_2} \right]$$

② 接着我们求误差 e 对路标点参数 $P_w = [x_w, y_w, z_w]^T$ 的偏导：

$$\frac{\partial e}{\partial P_w} = \frac{\partial e}{\partial P_c} \cdot \frac{\partial P_c}{\partial P_w}$$

由于 $P_c = \exp(\xi^{\wedge}) P_w = RP_w + t$，所以 $\frac{\partial P_c}{\partial P_w} = R$

于是 $\frac{\partial e}{\partial P_w} = \frac{\partial e}{\partial P_c} \cdot R$，

维度为 2×3

Jindong_Shi
2021-11-05

3. 直接法的 Bundle Adjustment

3.1 数学模型

1) 如何描述任意一点投影在任意一图像中形成的 error?

对于任意一个 3D 点 $p_i = [x_i\ \ y_i\ \ z_i]^T$ 投影至图像 j 的 error 可定义为该 3D 点经估计的变换矩阵T_j变换后，投影至图像 j 上，该投影点所在 4x4 大小的窗口内所有像素的灰度值$\begin{pmatrix} u = u_i - 2, \dots, u_i + 1 \\ v = v_i - 2, \dots, v_i + 1 \end{pmatrix}$，与该点对应给定的 4x4 窗口内的像素灰度值误差的平方和。

$$cost_i = \sum_{x=-2}^{1} \sum_{y=-2}^{1} \left\| \underbrace{I_{x,y}(p_i) - I_j\left(\pi(KT_jp_i) + \begin{pmatrix} x \\ y \end{pmatrix}\right)}_{e_{xy}(\xi_j, p_i)} \right\|^2$$

其中K为相机内参，π 为投影函数。$I_{x,y}(p_i)$ 代表给定的 3D 点 p_i 对应的 4x4 窗口内的像素灰度值。

每一个相机位姿 ξ_j和 3D 路标点的坐标位置 p_i之间均存在一条边，而这条边的误差项我们可以定义成 16 维，每一维存储一个 4x4 大小窗口内单个像素对应的误差$e_{xy}(\xi_j, p_i)$。

2) 每个 error 关联几个优化变量?

每个 error 关联两个优化变量，一个是对应每张图片的相机位姿 ξ_j，一个是所有 3D 点的坐标位置 p_i。

其中单个像素误差项具体为:

$$e_{xy}(\xi_j, p_i) = I_{x,y}(p_i) - I_j\left(\pi(KT_jp_i) + \begin{pmatrix} x \\ y \end{pmatrix}\right) = I_{x,y}(p_i) - I_j\left(\frac{1}{Z_{trans}} K \exp(\xi_j^\wedge)\, p_i + \begin{pmatrix} x \\ y \end{pmatrix}\right)$$

其中 Z_{trans} 为 3D 点 p_i 经变换矩阵 T_j 变换后的深度。这里ξ_j为相机位姿T_j的李代数。为了匹配 Sophus 中 se(3)的定义方式，我们将其设为平移在前，旋转在后: $\xi_j = [\rho_j\ \ \phi_j]^T$

3) 误差项关于各变量的雅可比是什么?

我们先讨论单个像素的误差项，再将 16 个单个像素的误差项合成一个误差项。

单个像素的误差项$e(\xi_j, p_i)$对于每张图片的相机位姿 ξ_j的雅可比维度为 1×6。

对误差项左乘扰动:

$$\begin{aligned} e_{xy}(\xi_j \oplus \delta\xi_j) &= I_{x,y}(p_i) - I_j\left(\frac{1}{Z_{trans}} K \exp(\delta\xi_j^\wedge) \exp(\xi_j^\wedge)\, p_i + \begin{pmatrix} x \\ y \end{pmatrix}\right) \\ &\approx I_{x,y}(p_i) - I_j\left(\frac{1}{Z_{trans}} K (1 + \delta\xi_j^\wedge) \exp(\xi_j^\wedge)\, p_i + \begin{pmatrix} x \\ y \end{pmatrix}\right) \\ &= I_{x,y}(p_i) - I_j\left(\frac{1}{Z_{trans}} K \exp(\xi_j^\wedge)\, p_i + \underbrace{\frac{1}{Z_{trans}} K\, \underbrace{\delta\xi_j^\wedge \exp(\xi_j^\wedge)\, p_i}_{q_{ij}}}_{u_{ij}} + \begin{pmatrix} x \\ y \end{pmatrix}\right) \end{aligned}$$

其中 $q_{ij} = [X, Y, Z]^T$ 为 p_i 经过 T_j 变换后的 3D 点坐标，u_{ij}为该点投影后的坐标，利用一阶泰勒展开:

$$e_{xy}(\xi_j \oplus \delta\xi_j) = \underbrace{I_{x,y}(p_i) - I_j\left(\frac{1}{Z_{cur}} K \exp(\xi_j^\wedge)\, p_i + \begin{pmatrix} x \\ y \end{pmatrix}\right)}_{e(\xi_j)} \underbrace{- \frac{\partial I_j}{\partial u_{ij}} \frac{\partial u_{ij}}{\partial q_{ij}} \frac{\partial q_{ij}}{\partial \delta\xi_j}}_{J} \delta\xi$$

其中 $\frac{\partial I_j}{\partial u_{ij}}$ 为当前图像在投影点 u_{ij} 处的梯度:

$$\frac{\partial I_j}{\partial u_{ij}} = \left(\frac{I_j(u_x+1, u_y) - I_j(u_x-1, u_y)}{2},\ \ \frac{I_j(u_x, u_y+1) - I_j(u_x, u_y-1)}{2}\right)$$

其中 $\frac{\partial u_{ij}}{\partial q_{ij}}$ 为投影方程关于相机坐标系下三维点$q_{ij} = [X, Y, Z]^T$的导数为:

$$\frac{\partial u_{ij}}{\partial q_{ij}} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} \end{bmatrix}$$

$\frac{\partial q_i}{\partial \delta\xi_j}$ 为变换后的三维点对变换的导数:

$$\frac{\partial q_{ij}}{\partial \delta \xi_j} = [I, \ -q_{ij}^{\wedge}] = \begin{bmatrix} 1 & 0 & 0 & 0 & Z & -Y \\ 0 & 1 & 0 & -Z & 0 & X \\ 0 & 0 & 1 & Y & -X & 0 \end{bmatrix}$$

合并 $\frac{\partial u_{ij}}{\partial q_{ij}}$ 和 $\frac{\partial q_{ij}}{\partial \delta \xi_j}$ 得到 $\frac{\partial u_{ij}}{\partial \delta \xi_j}$ 为:

$$\frac{\partial u_{ij}}{\partial \delta \xi_j} = \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{f_x X}{Z^2} & -\frac{f_x XY}{Z^2} & f_x + \frac{f_x X^2}{Z^2} & -\frac{f_y Y}{Z} \\ 0 & \frac{f_y}{Z} & -\frac{f_y Y}{Z^2} & -f_y - \frac{f_y Y^2}{Z^2} & \frac{f_y XY}{Z^2} & \frac{f_y X}{Z} \end{bmatrix}$$

最终得到的误差$e_{xy}(\xi_j, p_i)$相对于自变量李代数ξ_j的雅可比为:

$$J_{e\xi} = -\frac{\partial I_j}{\partial u_{ij}} \frac{\partial u_{ij}}{\partial \delta \xi_j}$$

该误差项对于 3D 点的坐标位置 p_i的雅可比 J_{ep} 维度为 1×3。

$$J_{ep} = -\frac{\partial I_j}{\partial u_{ij}} \frac{\partial u_{ij}}{\partial q_{ij}} \frac{\partial q_{ij}}{\partial p_i}$$

其中由于 $q_{ij} = R_j p_i + t_j$, 所以 $\frac{\partial q_{ij}}{\partial p_i} = R_j$, 我们得到雅可比 J_{ep}:

$$J_{ep} = -\frac{\partial I_j}{\partial u_{ij}} \frac{\partial u_{ij}}{\partial q_{ij}} R_j$$

最后我们将 16 个误差项 (按照下图序号的顺序)的雅可比排成一个竖列，分别组成 16x6 的 $J_{e\xi}$ 和 16x3 的 J_{ep}。

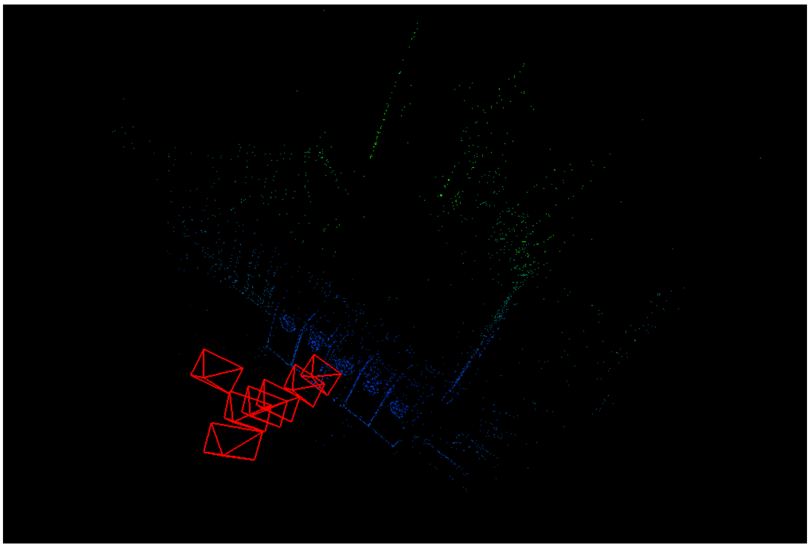
1	5	9	13
2	6	10	14
3	7	11	15
4	8	12	16

3.2 实现

源文件 directBA.cpp 位于文件夹 Direct_BA 内，可执行文件位于文件夹 Direct_BA/OUTPUT 内。由于下载的作业资料里面只包含了图片 1.png – 6.png，所以将图片 1.png 复制了一份，将其命名为 0.png。运行后发现 lambda 的值先从 370502 左右降到 24 左右，后又逐渐上升，运行结果和点云展示见下图。

```
jindong@jindong-virtual-machine:~/SLAM/Chap7/L7_code/Direct_BA/OUTPUT$ ./directBA
poses: 7, points: 4118
iteration= 0     chi2= 4155163.133872    time= 0.188488  cumTime= 0.188488      edges= 28826     schur= 1        lambda= 370502.930801   levenbergIter= 1
iteration= 1     chi2= 4028130.400817    time= 0.188961  cumTime= 0.377449      edges= 28826     schur= 1        lambda= 123500.976934   levenbergIter= 1
iteration= 2     chi2= 3929431.366723    time= 0.17127   cumTime= 0.548719      edges= 28826     schur= 1        lambda= 41166.992311    levenbergIter= 1
iteration= 3     chi2= 3859423.089369    time= 0.171739  cumTime= 0.720457      edges= 28826     schur= 1        lambda= 13722.330770    levenbergIter= 1
iteration= 4     chi2= 3802762.076494    time= 0.172698  cumTime= 0.893155      edges= 28826     schur= 1        lambda= 4574.110257     levenbergIter= 1
iteration= 5     chi2= 3774785.509439    time= 0.177315  cumTime= 1.07047       edges= 28826     schur= 1        lambda= 3049.406838     levenbergIter= 1
iteration= 6     chi2= 3734019.466040    time= 0.189754  cumTime= 1.26022       edges= 28826     schur= 1        lambda= 2032.937892     levenbergIter= 1
iteration= 7     chi2= 3703871.415347    time= 0.195862  cumTime= 1.45609       edges= 28826     schur= 1        lambda= 1355.291928     levenbergIter= 1
iteration= 8     chi2= 3682623.234952    time= 0.187663  cumTime= 1.64375       edges= 28826     schur= 1        lambda= 903.527952      levenbergIter= 1
iteration= 9     chi2= 3673277.872425    time= 0.197103  cumTime= 1.84085       edges= 28826     schur= 1        lambda= 602.351968      levenbergIter= 1
iteration= 10    chi2= 3654149.257729    time= 0.181143  cumTime= 2.02199       edges= 28826     schur= 1        lambda= 401.567979      levenbergIter= 1
iteration= 11    chi2= 3647657.814120    time= 0.193782  cumTime= 2.21578       edges= 28826     schur= 1        lambda= 267.711986      levenbergIter= 1
iteration= 12    chi2= 3645837.117134    time= 0.175963  cumTime= 2.39174       edges= 28826     schur= 1        lambda= 178.474657      levenbergIter= 1
iteration= 13    chi2= 3642708.023131    time= 0.177147  cumTime= 2.56889       edges= 28826     schur= 1        lambda= 118.983105      levenbergIter= 1
iteration= 14    chi2= 3634395.031623    time= 0.174536  cumTime= 2.74342       edges= 28826     schur= 1        lambda= 79.322070       levenbergIter= 1
iteration= 15    chi2= 3631219.291406    time= 0.172664  cumTime= 2.91609       edges= 28826     schur= 1        lambda= 52.881380       levenbergIter= 1
iteration= 16    chi2= 3626467.509911    time= 0.186261  cumTime= 3.10235       edges= 28826     schur= 1        lambda= 35.254253       levenbergIter= 1
iteration= 17    chi2= 3622448.376015    time= 0.17523   cumTime= 3.27758       edges= 28826     schur= 1        lambda= 23.502836       levenbergIter= 1
iteration= 18    chi2= 3621421.980701    time= 0.23072   cumTime= 3.5083        edges= 28826     schur= 1        lambda= 125.348456      levenbergIter= 3
iteration= 19    chi2= 3618791.460494    time= 0.171193  cumTime= 3.67949       edges= 28826     schur= 1        lambda= 83.565637       levenbergIter= 1
iteration= 20    chi2= 3618549.723815    time= 0.175965  cumTime= 3.85546       edges= 28826     schur= 1        lambda= 55.710425       levenbergIter= 1
iteration= 21    chi2= 3615203.970277    time= 0.188556  cumTime= 4.04401       edges= 28826     schur= 1        lambda= 37.140283       levenbergIter= 1
iteration= 22    chi2= 3609287.671619    time= 0.173088  cumTime= 4.2171        edges= 28826     schur= 1        lambda= 24.760189       levenbergIter= 1
iteration= 23    chi2= 3606038.880711    time= 0.253769  cumTime= 4.47087       edges= 28826     schur= 1        lambda= 1056.434724     levenbergIter= 4
iteration= 24    chi2= 3603794.608692    time= 0.175792  cumTime= 4.64666       edges= 28826     schur= 1        lambda= 704.289816      levenbergIter= 1
iteration= 25    chi2= 3601892.811231    time= 0.171912  cumTime= 4.81857       edges= 28826     schur= 1        lambda= 469.526544      levenbergIter= 1
iteration= 26    chi2= 3601687.330293    time= 0.244321  cumTime= 5.06289       edges= 28826     schur= 1        lambda= 2504.141569     levenbergIter= 3
iteration= 27    chi2= 3601419.625072    time= 0.199224  cumTime= 5.26212       edges= 28826     schur= 1        lambda= 3338.855425     levenbergIter= 2
iteration= 28    chi2= 3598242.214079    time= 0.179755  cumTime= 5.44187       edges= 28826     schur= 1        lambda= 2225.903617     levenbergIter= 1
iteration= 29    chi2= 3597832.834118    time= 0.204272  cumTime= 5.64614       edges= 28826     schur= 1        lambda= 1483.935745     levenbergIter= 1
iteration= 30    chi2= 3594783.697590    time= 0.180162  cumTime= 5.82631       edges= 28826     schur= 1        lambda= 989.290496      levenbergIter= 1
iteration= 31    chi2= 3587548.279610    time= 0.268364  cumTime= 6.09467       edges= 28826     schur= 1        lambda= 42209.727846    levenbergIter= 4
iteration= 32    chi2= 3581874.149830    time= 0.175312  cumTime= 6.26998       edges= 28826     schur= 1        lambda= 28139.818564    levenbergIter= 1
iteration= 33    chi2= 3580593.136506    time= 0.200182  cumTime= 6.47016       edges= 28826     schur= 1        lambda= 37519.758085    levenbergIter= 2
iteration= 34    chi2= 3579440.626636    time= 0.17464   cumTime= 6.6448        edges= 28826     schur= 1        lambda= 25013.172057    levenbergIter= 1
```

Jindong_Shi
2021-11-05



回答问题:

1) 我们经常用一个点的世界坐标 x,y,z 三个量来描述它，这是一种参数化形式。这里 x,y,z 三个量都是随机的，它们服从三维的高斯分布。而考虑到我们在相机看到某个点时，它的图像坐标 u,v 是比较确定的（u，v 的不确定性取决于图像的分辨率）, 而深度值 d 则是非常不确定的。所以我们考虑使用图像坐标 u,v 和深度值 d 来描述某个空间点。u,v 不动，而 d 服从（一维的）高斯分布。仿真发现，假设深度的倒数（也就是逆深度），为高斯分布是比较有效的。这就是所谓的逆深度参数法。[2]
2) 我们可以取稍大一些的 patch, 可以更好的区分不同的路标点，来增加计算的准确性，减少状态估计结果受到随机噪声带来的影响，但是加大 patch 同时也会迅速加大计算量。
3) 在 BA 阶段，特征点法需要给定路标点在对应相机位姿下的观测结果(重投影的像素坐标)，在给定相机位姿和路标点关联情况下，最小化重投影误差。直接法需要给定相机位姿和路标点的初始值，路标点的固定灰度值，无需给定相机位姿和路标点的关联的情况下，最小化光度误差。
4) 我们可以通过大量观测，取得误差项含有的随机误差的均值，这个随机误差可能是由测量噪声带来的。通过这个随机误差的均值来调整 Huber 的阈值。如果误差项的随机误差大于这个均值，则怀疑是误匹配造成的，通过 Huber 函数可以剔除误匹配对优化结果的影响。

参考文献:

[1] Bundle Ajustment - A Modern Synthesis, Bill Triggs, Philip Mclauchlan, Richard Hartley, Andrew Fitzgibbon
[2] SLAM 中的逆深度及参数化问题
https://blog.csdn.net/weixin_39568744/article/details/88582406