2. Bundle Adjustment

2.1 文献阅读

1. 为何说 Bundle Adjustment is slow 是不对的？
   说BA慢的大部分原因是没有考虑到问题本身的特点以及其稀疏性, 事实上使用合适的方法, BA速度可以很快.
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.
   BA的参数化空间基本上是一个非线性流形——一个由3D特征投影、3D旋转、和相机标定参数笛卡尔积组成的带有非线性约束的流形. 需要注意参数化空间的奇异性、内部约束(如四元素$\| q \|= 1$)、额外的自由度(如齐次坐标带有scale的自由度). 参数化方法对优化的速度、可靠性影响很大, 因此需要在当前估计附近寻找尽可能均匀, 有界, 表现良好的参数化方式.
   Point的参数化方式有①$(X,Y,Z)^T$ 和 ②$(X,Y,Z,W)^T$两种参数化方式. 参数化方式①下, cost function会很平滑, 距离越远需要的调整的步长越长, 若点的距离都很近则很好, 若有远距离点, 则参数化空间的均匀性会被严重破坏. 参数化方式②下, 对于远距离的点的处理则自然很多, 但需要添加额外约束$\sum X_i^2 = 1$.
   Pose参数化中, 对于旋转部分, 可以使用欧拉角、四元素、李代数. 欧拉角的话需要处理奇异性, 四元素需要归一化.
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.
   - 对于相机深度值的参数化方法, 现在流行使用逆深度. 相比于直接使用深度, 参数化空间更加均匀, 也能更好地表示远近不一的点.
   - 其中关于Network Structure的论述, 以及图结构稀疏性的分析, 即是后来的图优化.

2.2 BAL-dataset

投影:

$$
\begin{aligned}
\mathbf{q} &= \exp(\phi)\cdot\mathbf{p}+\mathbf{t} \\
\mathbf{q}' &= -\mathbf{q}/\mathbf{q}.z \\
\mathbf{u} &= f\cdot r(\mathbf{q}')\cdot\mathbf{q}' \\
r(\mathbf{q}') &= 1.0 + k1\cdot \| \mathbf{q}' \|^2 + k2\cdot \| \mathbf{q}' \|^4
\end{aligned}
$$

重投影误差:

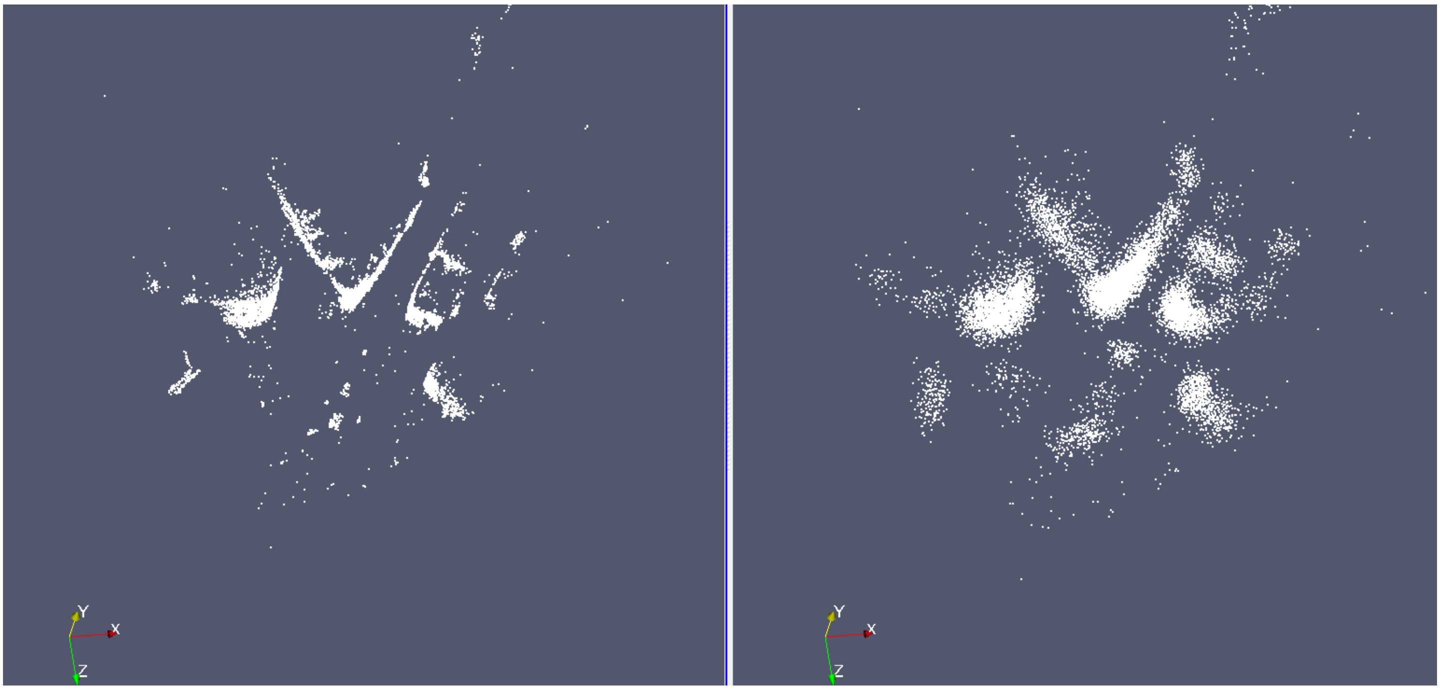
$$
\mathbf{e} = \mathbf{x}_{observe} - \mathbf{u}
$$

令$c = [f\ k1\ k2]^T$, 有Jacobian:

$$
\begin{aligned}
\frac{\partial \mathbf{e}}{\partial \mathbf{c}} &= [r(\mathbf{q}')\cdot\mathbf{q}' \quad f\cdot(\| \mathbf{q}' \|^2)\cdot\mathbf{q}' \quad f\cdot(\| \mathbf{q}' \|^4)\cdot\mathbf{q}'] \\
\frac{\partial \mathbf{e}}{\partial \mathbf{p}} &= -\frac{\partial \mathbf{u}}{\partial \mathbf{q}'}\frac{\partial \mathbf{q}'}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{p}} \\
\frac{\partial \mathbf{e}}{\partial \phi} &= -\frac{\partial \mathbf{u}}{\partial \mathbf{q}'}\frac{\partial \mathbf{q}'}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \phi} \\
\frac{\partial \mathbf{e}}{\partial \mathbf{t}} &= -\frac{\partial \mathbf{u}}{\partial \mathbf{q}'}\frac{\partial \mathbf{q}'}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{t}}
\end{aligned}
$$

程序运行结果(problem-21-11315-pre.txt):

```
ca_num=21 num_pts=11315 num_obs=36455
iteration= 0     chi2= 2309673.669349    time= 9.43278   cumTime= 9.43278        edges= 36455    schur= 1        lambda= 352.383325      levenbergIter= 1
iteration= 1     chi2= 152703.885345     time= 9.12934   cumTime= 18.5621        edges= 36455    schur= 1        lambda= 117.461108      levenbergIter= 1
iteration= 2     chi2= 51002.922181      time= 9.23198   cumTime= 27.7941        edges= 36455    schur= 1        lambda= 39.153703       levenbergIter= 1
iteration= 3     chi2= 40600.527358      time= 9.53807   cumTime= 37.3322        edges= 36455    schur= 1        lambda= 13.051234       levenbergIter= 1
iteration= 4     chi2= 38477.749569      time= 9.25078   cumTime= 46.583         edges= 36455    schur= 1        lambda= 4.350411        levenbergIter= 1
iteration= 5     chi2= 37826.676993      time= 9.66228   cumTime= 56.2452        edges= 36455    schur= 1        lambda= 2.345986        levenbergIter= 1
iteration= 6     chi2= 37644.113560      time= 9.56897   cumTime= 65.8142        edges= 36455    schur= 1        lambda= 1.563990        levenbergIter= 1
iteration= 7     chi2= 37225.357991      time= 9.48299   cumTime= 75.2972        edges= 36455    schur= 1        lambda= 1.042660        levenbergIter= 1
iteration= 8     chi2= 36898.537227      time= 9.07285   cumTime= 84.37  edges= 36455    schur= 1        lambda= 0.695107        levenbergIter= 1
iteration= 9     chi2= 36655.624967      time= 9.10672   cumTime= 93.4768        edges= 36455    schur= 1        lambda= 0.430000        levenbergIter= 1
iteration= 10    chi2= 36455.002430      time= 9.09878   cumTime= 102.576        edges= 36455    schur= 1        lambda= 0.143333        levenbergIter= 1
iteration= 11    chi2= 36093.472907      time= 9.30195   cumTime= 111.877        edges= 36455    schur= 1        lambda= 0.047778        levenbergIter= 1
iteration= 12    chi2= 35662.826436      time= 10.5192   cumTime= 122.397        edges= 36455    schur= 1        lambda= 0.057702        levenbergIter= 2
iteration= 13    chi2= 35296.904329      time= 9.49559   cumTime= 131.892        edges= 36455    schur= 1        lambda= 0.038468        levenbergIter= 1
iteration= 14    chi2= 35287.433858      time= 9.72328   cumTime= 141.616        edges= 36455    schur= 1        lambda= 0.025646        levenbergIter= 1
iteration= 15    chi2= 33788.151378      time= 10.7609   cumTime= 152.376        edges= 36455    schur= 1        lambda= 0.029491        levenbergIter= 2
iteration= 16    chi2= 33086.340597      time= 10.4462   cumTime= 162.823        edges= 36455    schur= 1        lambda= 0.034192        levenbergIter= 2
iteration= 17    chi2= 32924.254293      time= 9.0683    cumTime= 171.891        edges= 36455    schur= 1        lambda= 0.022795        levenbergIter= 1
iteration= 18    chi2= 31978.946733      time= 10.1318   cumTime= 182.023        edges= 36455    schur= 1        lambda= 0.022872        levenbergIter= 2
iteration= 19    chi2= 31508.447798      time= 10.116    cumTime= 192.139        edges= 36455    schur= 1        lambda= 0.030496        levenbergIter= 2
iteration= 20    chi2= 31230.834049      time= 9.03642   cumTime= 201.175        edges= 36455    schur= 1        lambda= 0.020330        levenbergIter= 1
iteration= 21    chi2= 31125.288534      time= 9.11369   cumTime= 210.289        edges= 36455    schur= 1        lambda= 0.013554        levenbergIter= 1
iteration= 22    chi2= 30226.384058      time= 10.0976   cumTime= 220.386        edges= 36455    schur= 1        lambda= 0.012962        levenbergIter= 2
iteration= 23    chi2= 29887.246669      time= 10.1012   cumTime= 230.488        edges= 36455    schur= 1        lambda= 0.017283        levenbergIter= 2
iteration= 24    chi2= 29744.647646      time= 9.09018   cumTime= 239.578        edges= 36455    schur= 1        lambda= 0.011522        levenbergIter= 1
iteration= 25    chi2= 29738.378210      time= 9.07032   cumTime= 248.648        edges= 36455    schur= 1        lambda= 0.007681        levenbergIter= 1
iteration= 26    chi2= 29114.020083      time= 10.1217   cumTime= 258.77         edges= 36455    schur= 1        lambda= 0.007955        levenbergIter= 2
iteration= 27    chi2= 28934.776002      time= 10.0964   cumTime= 268.866        edges= 36455    schur= 1        lambda= 0.010606        levenbergIter= 2
iteration= 28    chi2= 28884.571756      time= 9.09775   cumTime= 277.964        edges= 36455    schur= 1        lambda= 0.007071        levenbergIter= 1
iteration= 29    chi2= 28677.735247      time= 10.1465   cumTime= 288.111        edges= 36455    schur= 1        lambda= 0.007393        levenbergIter= 2
iteration= 30    chi2= 28593.759491      time= 10.1037   cumTime= 298.214        edges= 36455    schur= 1        lambda= 0.009857        levenbergIter= 2
iteration= 31    chi2= 28543.773519      time= 9.10365   cumTime= 307.318        edges= 36455    schur= 1        lambda= 0.006571        levenbergIter= 1
iteration= 32    chi2= 28537.953596      time= 9.08204   cumTime= 316.4  edges= 36455    schur= 1        lambda= 0.004381        levenbergIter= 1
iteration= 33    chi2= 28368.711530      time= 10.1612   cumTime= 326.561        edges= 36455    schur= 1        lambda= 0.005519        levenbergIter= 2
iteration= 34    chi2= 28299.780396      time= 10.117    cumTime= 336.678        edges= 36455    schur= 1        lambda= 0.005727        levenbergIter= 2
iteration= 35    chi2= 28256.565563      time= 10.1905   cumTime= 346.869        edges= 36455    schur= 1        lambda= 0.006810        levenbergIter= 2
iteration= 36    chi2= 28235.733669      time= 9.09262   cumTime= 355.961        edges= 36455    schur= 1        lambda= 0.004540        levenbergIter= 1
iteration= 37    chi2= 28167.339034      time= 10.0674   cumTime= 366.029        edges= 36455    schur= 1        lambda= 0.005149        levenbergIter= 2
iteration= 38    chi2= 28129.877056      time= 10.1205   cumTime= 376.149        edges= 36455    schur= 1        lambda= 0.005605        levenbergIter= 2
iteration= 39    chi2= 28118.568397      time= 9.1303    cumTime= 385.279        edges= 36455    schur= 1        lambda= 0.003737        levenbergIter= 1
```



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?
   任意一点在任意一图像中形成的error与直接法一样: 为投影后整个patch与对应图像上的patch的光度误差.

$$e = I(p_i) - I_j(\pi(KT_jp_i))$$

2. 每个error关联几个优化变量?
   每个error关联优化变量: 相机位姿和3D点的坐标.
3. error关于各变量的雅可比是什么?
   令$q = T(\xi)p$,$u = \pi(Kq)$

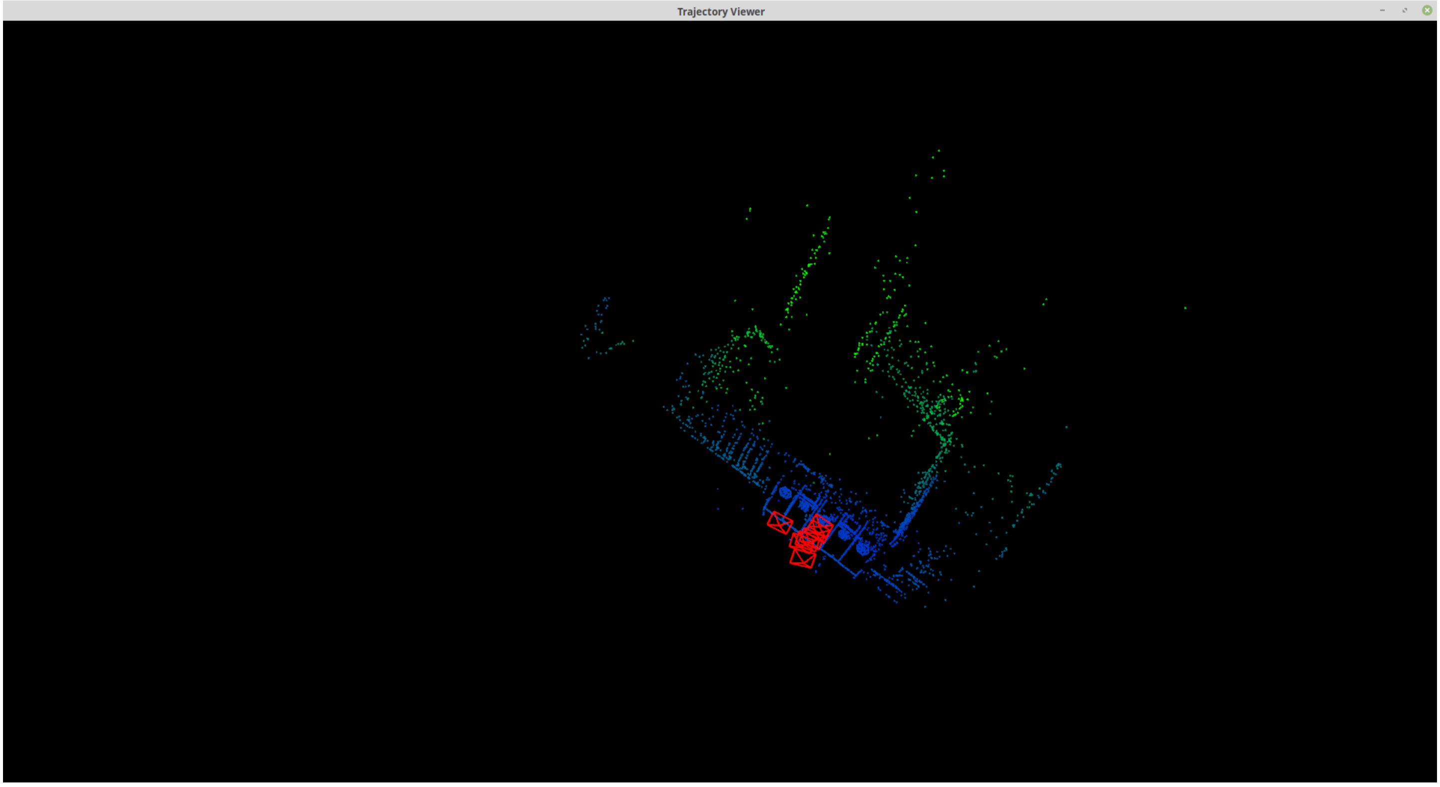
$$\frac{\partial e}{\partial p} = -\frac{\partial I}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \mathbf{p}}$$
$$\frac{\partial e}{\partial \xi} = -\frac{\partial I}{\partial \mathbf{u}}\frac{\partial \mathbf{u}}{\partial \mathbf{q}}\frac{\partial \mathbf{q}}{\partial \xi}$$

这里

$$
\begin{aligned}
\mathbf{u} &= \frac{1}{Z}\begin{bmatrix} fx & 0 & cx \\ 0 & f_y & cy \end{bmatrix}\mathbf{p} \\
\Rightarrow \frac{\partial \mathbf{u}}{\partial \mathbf{q}} &= \begin{bmatrix} \frac{f_x}{Z} & 0 & -\frac{xf_x}{Z^2} \\ 0 & \frac{f_y}{Z} & -\frac{yf_y}{Z^2} \end{bmatrix} \\
\frac{\partial \mathbf{q}}{\partial \xi} &= [I \ \ -\mathbf{q}^{\wedge}] \\
\frac{\partial \mathbf{q}}{\partial \mathbf{p}} &= T(\xi)
\end{aligned}
$$

3.2实现

程序运行效果:



1. 能否不要以 $[x, y, z]^T$ 的形式参数化每个点?
   可以以$[x, y, z_{inv}]^T$的方式参数化每个点, $z_{inv} = \frac{1}{z}$.
2. 取 4x4 的 patch 好吗?取更大的 patch 好还是取小一点的 patch 好?
   取4x4的patch好, 这个问题和之前直接法计算相机位姿一样, 当patch取小了, 没有区分性. 而patch取大了则会加入很多outlier.
3. 由于图像的差异,你可能需要鲁棒核函数,例如 Huber. 此时Huber的阈值如何选取?
   分析投影后的误差, 根据误差以及匹配情况， 分析匹配和不匹配的分界, 选取Huber的阈值.