深蓝学院

视觉 SLAM

第四章作业

Student name: *MatthewNUST*

Course: *视觉 SLAM 理论与实践* – Professor: *高翔*

Due date: *October 18th, 2020*

2. 图像去畸变

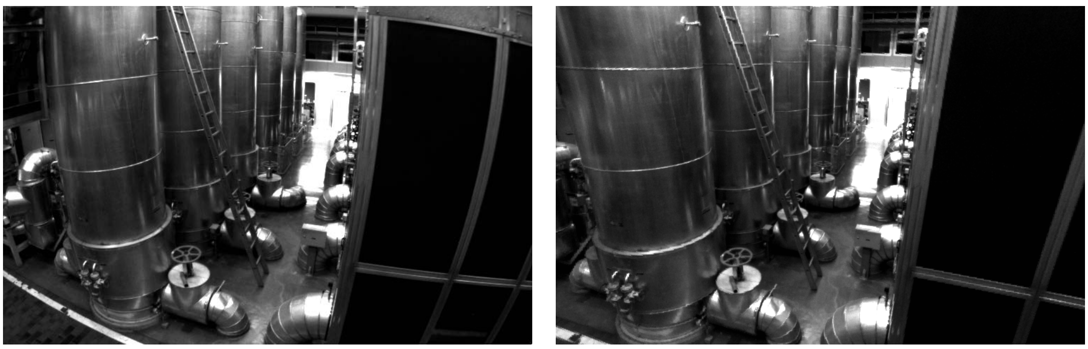
请根据所给文件中内容，完成对该图像的去畸变操作。

Answer. 基本思路是先将像素平面坐标变换到成像平面（归一化平面）上进行畸变纠正，纠正完毕后再变换回像素平面。添加的代码部分如下。

```
double x = (u-cx)/fx, y = (v - cy)/fy;  //变换到成像平面坐标系XOY
double rpow2 = pow(x, 2) + pow(y, 2);
double x_distorted = x*(1+k1*rpow2 + k2*rpow2*rpow2) + 2*p1*x*y + p2*(rpow2 + 2*x*x);
double y_distorted = y*(1+k1*rpow2 + k2*rpow2*rpow2) + 2*p2*x*y + p1*(rpow2 + 2*y*y);
u_distorted = fx * x_distorted + cx;
v_distorted = fy * y_distorted + cy;
```

Listing 1: 题 2 所添代码

得到的去畸变后的图像如图 1(b) 所示，与原图进行对比可知去畸变后图像在边缘存在一定损失。



(a) 测试图像 (b) 去畸变后图像

图 1: 去畸变前后对比

3. 双目视差的使用

> 请根据给定参数，计算相机数据对应的点云，并显示到 Pangolin 中。程序请参考 code/disparity.cpp 文件。

Answer. 在程序中添加上以下几行代码即可得到图 2 结果。

```
point[2] = fx*d/disparity.at<uchar>(v,u);
point[0] = (u-cx)*point[2]/fx;
point[1] = (v-cy)*point[2]/fy;
pointcloud.push_back(point);
```

Listing 2: 题 3 所添代码

值得注意的是此处得到的点云结果是基于 left.png 的（相当于直接将 left.png 这一像素平面坐标变换到成像平面并基于视差数据并添加上深度这一维度），所以 disparity.cpp 中并未对 right.png 进行有用的操作，那么我们也可以基于 right.png 文件进行相关操作。

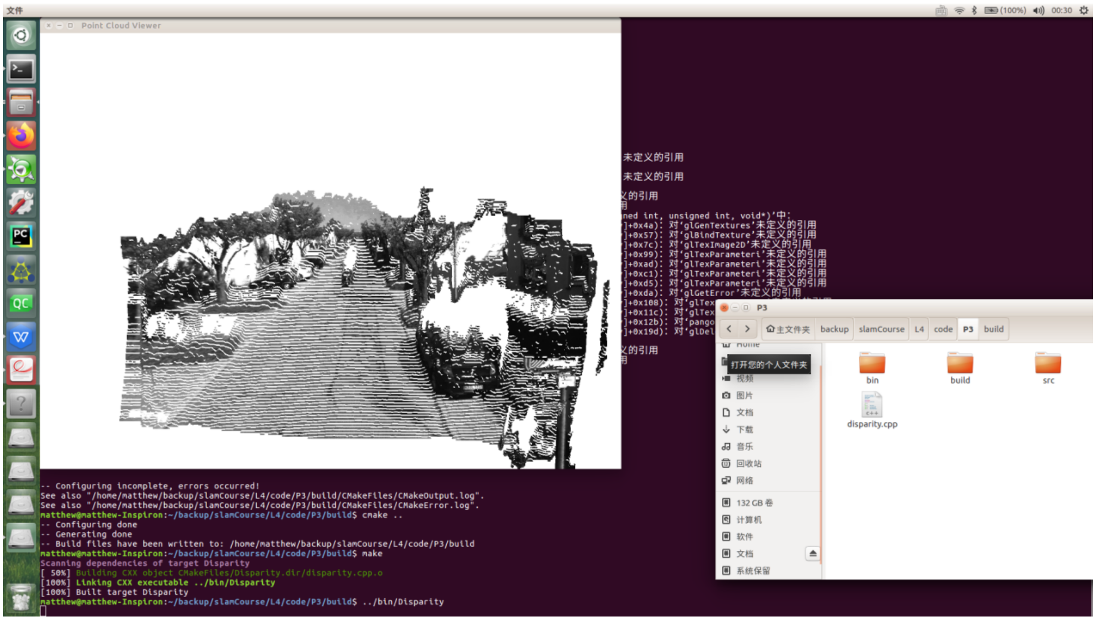


图 2: 运行结果

4. 矩阵运算微分

设变量为 $x \in R^N$，那么：

1. 矩阵 $A \in R^{N\times N}$，那么 d(Ax)/dx 是什么？

2. 矩阵 $A \in R^{N\times N}$，那么 d(xTAx)/dx 是什么?

3. 证明 $xTAx = tr(AxxT)$

Answer.

1. 本题我采用的是原始定义证明法则。

解 设 $A=(a_1, \cdots a_n)$ $x=(x_1, \cdots x_n)^T$ 其中a_i为n维列向量.

则AX为n×1的n维列向量 $AX=(a_1x_1, \cdots, a_nx_n)^T$

$$\therefore \frac{d(AX)}{dx} = \nabla_x(AX) = \left[\frac{\partial(AX)}{\partial x_1}, \cdots, \frac{\partial(AX)}{\partial x_n}\right]^T$$

$$= \left[\frac{\partial[(a_1x_1, \cdots a_nx_n)^T]}{\partial x_1}, \cdots, \frac{\partial[(a_1x_1, \cdots, a_nx_n)^T]}{\partial x_n}\right]^T$$

$$= [a_1, \cdots, a_n]^T$$

$$= A^T$$

2. 第一问使用了比较原始的证明方法，对于第二题而言证明过程会相当繁琐，所以在解答第二问之前我进一步查阅了相关资料 [1]，了解到存在更为简便的求解方法[1]，下面使用此方法进行证明，为节省时间使用了书写版。

[1]详见文献一 p277。

解：① $d(x^TAX) = d(x^TA)X + x^TAdX$

$= (dx^T A + x^T dA)X + x^TAdx$

由于A为常矩阵，故有

$d(x^TAX) = dx^TAX + x^TAdX$

② 对上式两边取迹，因 $d(x^TAX)$ 为标量，故取迹后仍为本身，则有

$d(x^TAX) = tr[dx^TAX + x^TA \cdot dx]$

$= tr(dx^TAX) + tr(x^TA \cdot dx)$

$= tr[(AX)^T dX]^T + tr(x^TAdX)$

$= tr[(x^TA^T + x^TA)dx]$

$= tr[x^T(A^T + A)dx]$

$= tr[[(A+A^T)X]^T dX]$

③ 据所述求解方法，则有 $\frac{d(x^TAX)}{dX} = (A+A^T)X$。此结果亦与文献一吻合

3. 证明如下

$$tr(AXX^T) = tr\left(\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \cdot [x_1, \cdots, x_n]\right)$$

$$= tr\left(\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1x_1 & \cdots & x_1x_n \\ \vdots & \ddots & \vdots \\ x_nx_1 & \cdots & x_nx_n \end{bmatrix}\right)$$

$$= \sum_{i=1}^{n} a_{1i}x_ix_1 + \sum_{i=1}^{n} a_{2i}x_ix_2 + \cdots + \sum_{i=1}^{n} a_{ni}x_ix_n$$

而 $$x^TAX = [x_1, \cdots x_n] \begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix}$$

$$= [x_1, \cdots, x_n] \begin{bmatrix} \sum_{i=1}^{n} x_i a_{1i} \\ \vdots \\ \sum_{i=1}^{n} x_i a_{ni} \end{bmatrix}$$

$= tr(AXX^T)$ 得证.

5. 高斯牛顿法的曲线拟合实验 (bonus marks)

现在请你书写 Gauss-Newton 的程序以解决此问题。程序框架见 code/gaussnewton.cpp，请填写程序内容以完成作业

Answer. 所添加的第一处代码如下所示。Error 代表第 i 个数据点的误差值，J[0]、J[1]、J[2] 是 error 分别关于当前估计值对 ae、be、ce 的倒数。

```
double error;   // 第i个数据点的计算误差
// 填写计算error的表达式
error = yi - exp( ae * xi * xi + be * xi + ce );
// 雅可比矩阵
Vector3d J;
J[0] = - xi * xi * exp(ae * xi * xi + be * xi + ce);  // de/da
J[1] = - xi * exp(ae * xi * xi + be * xi + ce);  // de/db
J[2] = - exp(ae * xi * xi + be * xi + ce);  // de/dc

H += J * J.transpose(); // GN近似的H
b += -error * J;
```

Listing 3: 题 5 所添代码

在求解线性方程时，由于 H 是雅各比矩阵乘以其自身的转置，故而 H 是一个 SPD（实对称正定阵），联想到第二章所做的作业 H 可以进行比 QR 分解更为高效的 Cholesky 分解，因此使用 Eigen 的 ldlt() 函数进行计算（可见高博出的题还是很环环相扣的），代码为：

```
Vector3d dx = H.ldlt().solve(b);
```

Listing 4: 题 5 所添代码

最后计算得到结果如图 3 所示，与所给参考答案一致。

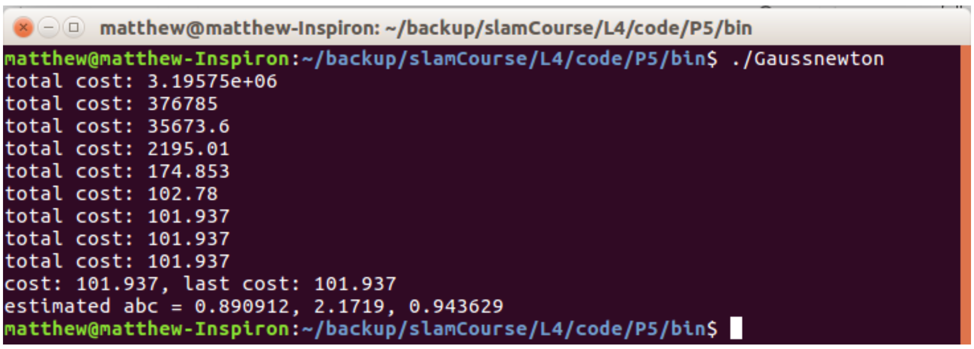


图 3: 运行结果

6.* 批量最大似然估计

1. 可以定义矩阵 H，使得批量误差为 $e = z - Hx$。请给出此处 H 的具体形式。
2. 请给出此问题下 W 的具体取值。
3. 假设所有噪声相互无关，该问题存在唯一的解吗？若有，唯一解是什么？若没有，说明理由。

Answer.

1. 证明如下：

6.1 据式(6.11) $e_{v,k} = x_k - f(x_{k-1}, u_k)$ ， $e_{y,j,k} = z_{k,j} - h(x_k, y_j)$.

对于运动误差，$e_1 = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} - \begin{pmatrix} h_1 \\ h_2 \\ h_3 \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$ ---- h_i 为 4×1 的行向量，$h_i = [l_{i1}, l_{i2}, l_{i3}, l_{i4}]$

$= \begin{pmatrix} v_1 - h_1 X \\ v_2 - h_2 X \\ v_3 - h_3 X \end{pmatrix}$

$= -\begin{pmatrix} h_1 X - v_1 \\ h_2 X - v_2 \\ h_3 X - v_3 \end{pmatrix}$ ---- 取反，使形式与 $x_k - f(x_{k-1}, v_k)$ 接近.

易知，我们欲得到 $h_i X - v_i = x_i - x_{i-1} - v_i$ 的形式，则 $h_1 = [-1\ 1\ 0\ 0]$

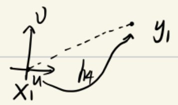
同理可得 $h_2 = [0\ -1\ 1\ 0]$ ， $h_3 = [0\ 0\ -1\ 1]$

$\therefore H_1 = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix}$ ，下面求解 H_2 部分.

$e_2 = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} - \begin{pmatrix} h_4 \\ h_5 \\ h_6 \end{pmatrix} X = \begin{pmatrix} y_1 - h_4 X \\ y_2 - h_5 X \\ y_3 - h_6 X \end{pmatrix}$

注意到 y_1 代表1时的路标，h_4 代表在 X_1 的坐标系下到 y_1 的变换、令 $h_4 = [T_{41}, T_{42}, T_{43}, T_{44}]$

$\because y_1 - h_4 X$ 代表了观测的误差，$\Rightarrow y_1 - h_4 X = y_1 - T_{42} x_1$，有 $h_4 = [0\ T_{42}\ 0\ 0]$



同理有 $h_5 = [0\ 0\ T_{54}\ 0]$ ， $h_6 = [0\ 0\ 0\ T_{64}]$ ，T_{ij} 为变换表示

故综上，$H = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & T_{42} & 0 & 0 \\ 0 & 0 & T_{53} & 0 \\ 0 & 0 & 0 & T_{63} \end{bmatrix}$

2. 证明如下：

6.2 对于本题，z可看作高斯分布 $z \sim N(Hx, W)$，下面求W的最大似然估计

$$p(z)=\frac{1}{\sqrt{(2\pi)^N \det(W)}}\exp\left(-\frac{1}{2}(z-Hx)^T W^{-1}(z-Hx)\right)$$ 取负对数，有

$$-\ln(p(z)) = \frac{1}{2}\ln[(2\pi)^N \det(W)] + \frac{1}{2}(z-Hx)^T W^{-1}(z-Hx)$$ 对两边取对W的微分有

$$-d[\ln(p(z))] = \frac{1}{2}\cdot\frac{(2\pi)^N d[\det(W)]}{(2\pi)^N \det(W)} + \frac{1}{2} d[(z-Hx)^T W^{-1}(z-Hx)]$$

$$= \frac{1}{2}\left\{tr(W^{-1}dW) + d[(z-Hx)^T W^{-1}(z-Hx)]\right\} \quad \text{---} \quad d(|A|) = |A|\, tr(A^{-1}dA)$$

题4中已证得 $x^TAx = tr(Axx^T)$，则有

$x^TA^{-1}x = tr(A^{-1}xx^T)$ 两边同时对A取微分

$$d(x^TA^{-1}x) = d[tr(A^{-1}xx^T)]$$

$$= tr[d(A^{-1}xx^T)] \quad \text{----} \quad d[tr(A)] = tr[d(A)]$$

$$= tr[dA^{-1}\cdot xx^T + A^{-1}d(xx^T)]$$

$$= tr(dA^{-1}xx^T) \quad \text{---} \quad d(xx^T)=0$$

$$= tr[-A^{-1}dA\cdot A^{-1}\cdot xx^T]$$

$$= tr[-A^{-1}xx^TA^{-1}dA] \quad \text{----} \quad tr(ABCD) = tr(CDAB)$$

$$\therefore -d[\ln p(z)] = \frac{1}{2} tr[(W^{-1} - W^{-1}(z-Hx)(z-Hx)^T W^{-1})dW]$$

则 $\frac{-d[\ln p(z)]}{dW} = \frac{1}{2}[W^{-1} - W^{-1}(z-Hx)(z-Hx)^T W^{-1}]^T$ 令其为0，则有

$$W^{-1} = W^{-1}(z-Hx)(z-Hx)^T W^{-1}$$

$W = (z-Hx)(z-Hx)^T$，此即该问题下W的具体取值

3. 该问题存在唯一解，唯一解为对第二问最小二乘问题求解所得到的解。

参考文献

[1] 张贤达. 矩阵分析与应用. 清华大学出版社，2004.

[2] 知乎." 三步搞定矩阵求导".https:https://zhuanlan.zhihu.com/p/262751195.

[3] CSDN." 机器人所涉及的相关数学理论整理".https://blog.csdn.net/GFDYVJG/article/details/

[4] "Relationship between the Hessian and Covariance Matrix for Gaussian Random Variables".https://onlinelibrary.wiley.com/doi/pdf/10.1002/9780470824566.app1