

# 第四章作业

Student name: Francisrk

Due date: February 20th, 2022

## 1 第 1 题

已阅。

## 2 第 2 题

在 undistort\_image.cpp 中加入以下代码：

```
1     double X=0, Y=0, X_distorted=0, Y_distorted=0, r=0;
2     X = (u - cx)/fx;
3     Y = (v - cy)/fy;
4     r = sqrt(X * X + Y * Y);
5     X_distorted = X * (1 + k1 * r * r + k2 * r * r * r * r) + 2 * p1
* X * Y + p2 * (r * r + 2 * X * X);
6     Y_distorted = Y * (1 + k1 * r * r + k2 * r * r * r * r) + p1 * (
r * r + 2 * Y * Y) + 2 * p2 * X * Y;
7     u_distorted = fx * X_distorted + cx;
8     v_distorted = fy * Y_distorted + cy;
```

Listing 1: 在 undistort\_image.cpp

```
1 cmake_minimum_required(VERSION 3.21)
2 project(T_1)
3 set(CMAKE_CXX_STANDARD 11)
4 find_package(OpenCV 3 REQUIRED)
5 add_executable(undistort_image undistort_image.cpp )
6 target_link_libraries(undistort_image ${OpenCV_LIBS})
```

Listing 2: 工程/CMakeLists.txt

去畸变前后的对比图如图 2.1 所示。



图 2.1 去畸变前后对比图

### 3 第 3 题

#### 3.1 请说明鱼眼相机相比于普通针孔相机在 SLAM 方面的优势

鱼眼相机最重要的一个优势就是相比于普通针孔相机拥有更宽阔的视野。因此可以确保在一段时间里，尽可能多的视觉特征进入相机视野，从而提高对周围环境的感知能力。

#### 3.2 请整理并描述 OpenCV 中使用的鱼眼畸变模型（等距投影） 是如何定义的，它与上题的畸变模型有何不同

本部分参考 [1]，鱼眼镜头一般是由十几个不同的透镜组合而成的，如下图所示，在成像的过程中，入射光线经过不同程度的折射，投影到尺寸有限的成像平面上，使得鱼眼镜头与普通镜头相比起来拥有了更大的视野范围。

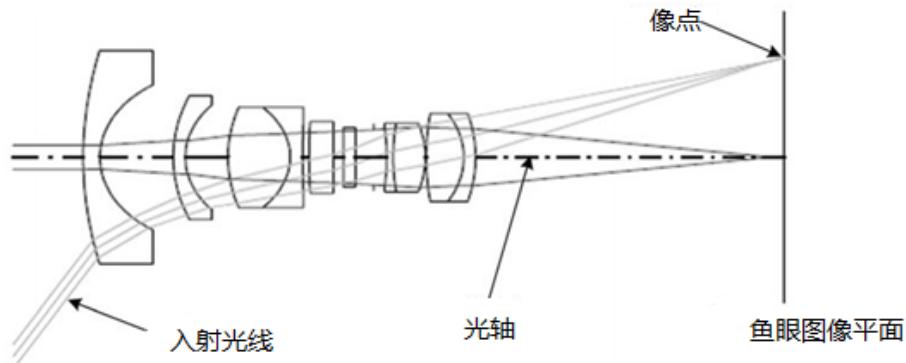


图 3.1 鱼眼镜头原理图

在研究鱼眼相机成像时，可以将上面的镜头组简化为一个球面，如图 3.2(b) 所示， $O_1 - X_cY_cZ_c$  是相机坐标系， $O_2 - xy$  是成像平面。现实世界有一点  $P$ ，入射角为  $\theta$ ，如果按照普通相机的针孔相机模型，入射光线  $PO_1$  经过镜头后不改变路线， $P, O_1, p'$  三点共线， $p'$  为  $P$  的像；但是对于鱼眼相机，入射光线  $PO_1$  经过镜头后会发生折射，因此  $P$  的像点为  $p$  点，极坐标为  $(r, \varphi)$ 。

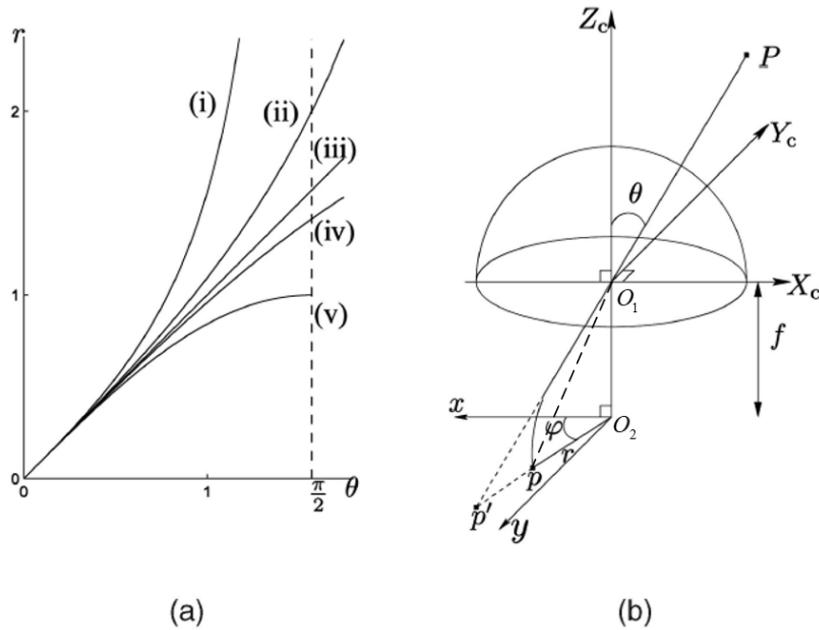


Fig. 1. (a) Projections (1), (2), (3), (4), and (5) with  $f = 1$ . (b) Fish-eye camera model. The image of the point  $P$  is  $p$  whereas it would be  $p'$  by a pinhole camera.

图 3.2 成像原理图 \_1

为了将尽可能大的场景投影到有限的图像平面内，鱼眼相机会按照一定的投影函数来设计，如图 3.2(a) 所示。根据投影函数的不同，鱼眼相机的设计模型大致能被分为五种：透视投影（即针孔相机模型）、等积投影、等距投影、体视投影、正交投影。

投影模型	投影函数	特征
i. 透视投影 (perspective projection)	$r = f \tan \theta$	针孔相机模型
ii. 体视投影 (stereographic projection)	$r = 2f \tan \frac{\theta}{2}$	任何直线相交的角度，在变换后保持不变
iii. 等距投影 (equidistance projection)	$r = f\theta$	物体成像面上距离画面中心的距离与入射角成正比
iv. 等积投影 (equisolid angle projection)	$r = 2f \sin \frac{\theta}{2}$	在变换前后，物体所占的立体角大小不变
v. 正交投影 (orthogonal projection)	$r = f \sin \theta$	投影畸变最大，而且最大视场角不能大于180°

图 3.3 鱼眼模型投影分类

接下来介绍鱼眼镜头的成像过程：实际的镜头因为各种原因并不会精确的符合投影模型，为了方便鱼眼相机的标定，一般取  $r$  关于  $\theta$  泰勒展

开式的前 5 项来近似鱼眼镜头的实际投影函数:

$$r(\theta) \approx k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 \quad (3.1)$$

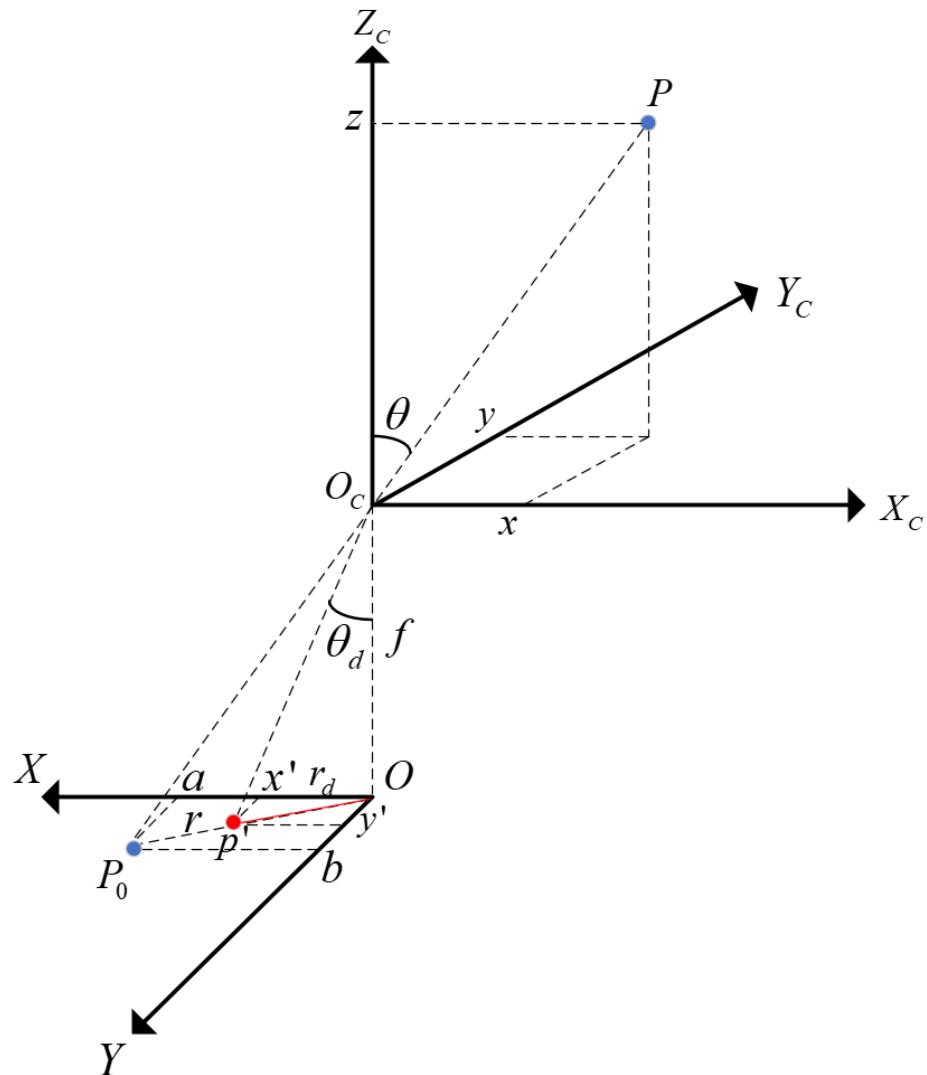


图 3.4 鱼眼模型成像过程

设相机系下有一点  $P(x, y, z)$ , 点  $P(x, y, z)$  如果按照针孔相机模型投影, 则不存在畸变, 像点为  $P_0(a, b)$ , 不妨设  $f = 1$  (最终可以求得  $\frac{r_d}{r}$  与  $f$

无关), 可得  $P_0$

$$\begin{cases} a = \frac{x}{z} \\ b = \frac{y}{z} \\ r^2 = a^2 + b^2 \\ \theta = \arctan(r) \end{cases} \quad (3.2)$$

由于畸变的存在,  $P_0$  点畸变成  $p'$  点,  $r$  被压缩到  $r_d$ , 实际的像点位置为  $p'(x', y')$ , 有  $|Op'| = r_d, |OP_0| = r$ , 结合等距离投影函数

$$r(\theta) = f(\theta) \approx k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 \quad (3.3)$$

其中  $r^2 = a^2 + b^2$  表示相机空间任一点  $P$  归一化平面上的像点  $p$  距离光心的距离  $|OP_0|$ , 根据对顶角  $\angle P_0 O_C O = \theta$ , 且有

$$\begin{cases} \tan(\theta) = \frac{r}{f} = r \\ \tan(\theta_d) = \frac{r_d}{f} = r_d \end{cases} \quad (3.4)$$

考虑到相机的成像 CCD 平面尺寸一般都是几毫米, 焦距在几百毫米左右, 所以相机实际成像过程中  $\theta_d$  通常很小,  $k_0$  可以取 1, 所以有

$$\begin{aligned} \tan(\theta_d) &\approx \theta_d = r_d = k_0\theta + k_1\theta^3 + k_2\theta^5 + k_3\theta^7 + k_4\theta^9 \\ &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \end{aligned} \quad (3.5)$$

如图 3.4, 畸变后的坐标为  $p'[x', y']$ , 由相似关系:

$$\frac{\theta_d}{r} = \frac{r_d}{r} = \frac{x'}{a} = \frac{y'}{b} = scale \quad (3.6)$$

所以有

$$\begin{cases} x' = scale * a \\ y' = scale * b \end{cases} \quad (3.7)$$

再根据相机内参转换成像素坐标  $[u, v]$

$$\begin{cases} u = f_x * x' + c_x \\ v = f_y * y' + c_y \end{cases} \quad (3.8)$$

有的地方还说有个  $\alpha$  偏度系数, 我不知道这个是怎么得来的。

小结: 由像素坐标计算未畸变的归一化坐标-> 找到畸变后的坐标-> 转换到像素平面。

更进一步：鱼眼相机的成像过程是已知入射角  $\theta$  求出射角  $\theta_d$ ，而鱼眼相机的畸变矫正则是已知畸变后的像点位置  $(x', y')$ ，求实际的入射角  $\theta$ 。由于相机参数已知，可以根据  $(x', y')$  以及相机焦距求得  $\theta_d$  的值，所以畸变矫正的本质问题是求解关于  $\theta$  的一元高次方程：

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \quad (3.9)$$

其中  $k_1, k_2, k_3, k_4$  是畸变系数，由相机标定结果提供。常用的求解一元高次方程的方法有二分法、不动点迭代、牛顿迭代法。如使用牛顿法迭代求解：

$$\begin{aligned} f(\theta) &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) - \theta_d \\ \theta_0 &= \theta_d \\ \theta_{n+1} &= \theta_n - \frac{f(\theta_n)}{f'(\theta_n)} \end{aligned} \quad (3.10)$$

### 3.3 完成 fisheye.cpp 文件中的内容。针对给定的图像，实现它的畸变校正。要求：通过手写方式实现，不允许调用 OpenCV 的 API

核心代码如下所示：

```

1 // 变到相机坐标系
2     double x = (u - cx) / fx, y = (v - cy) / fy;
3     double r = sqrt(x * x + y * y);
4     double theta = atan(r);
5     double theta_d = theta * (1 + k1 * pow(theta, 2) + k2 * pow(
theta, 4) + k3 * pow(theta, 6) + k4 * pow(theta, 8));
6     double scale = theta_d / r;
7     double x_distorted = scale * x;
8     double y_distorted = scale * y;
9     // 去畸变完之后再变换到像素
10    double u_distorted = fx * x_distorted + cx;
11    double v_distorted = fy * y_distorted + cy;

```

Listing 3: 工程/CMakeLists.txt



图 3.5 鱼眼模型去畸变之后的结果

**3.4 为什么在这张图像中，我们令畸变参数  $k_1, \dots, k_4 = 0$ ，依然可以产生去畸变的效果？**

由投影函数

$$\theta_d = \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \quad (3.11)$$

取泰勒展开前五项来近似鱼眼模型， $k_1 \dots k_4$  取 0，相当于只近似了第一项， $\theta_d$  的一次向系数  $k_0$  可以取 1，所以说仍然可以完成去畸变的操作。

**3.5 在鱼眼模型中，去畸变是否带来了图像内容的损失？如何避免这种图像内容上的损失呢？**

鱼眼图一般为圆形，边缘的信息被压缩的很密，经过去除畸变后原图中间的部分会被保留的很好，而边缘位置一般都会被拉伸的很严重、视觉效果差，所以通常会进行切除，因此肯定会带来图像内容的损失。增大去畸变时

图像的尺寸，或者使用单目相机和鱼眼相机图像进行融合，补全丢失的信息。

## 4 第 4 题

理论部分：

**4.1 推导双目相机模型下，视差与 XYZ 坐标的关系式。请给出由像素坐标加视差  $u, v, d$  推导  $XYZ$  与已知  $XYZ$  推导  $u, v, d$  两个关系。**

双目相机成像模型如图 4.1 所示

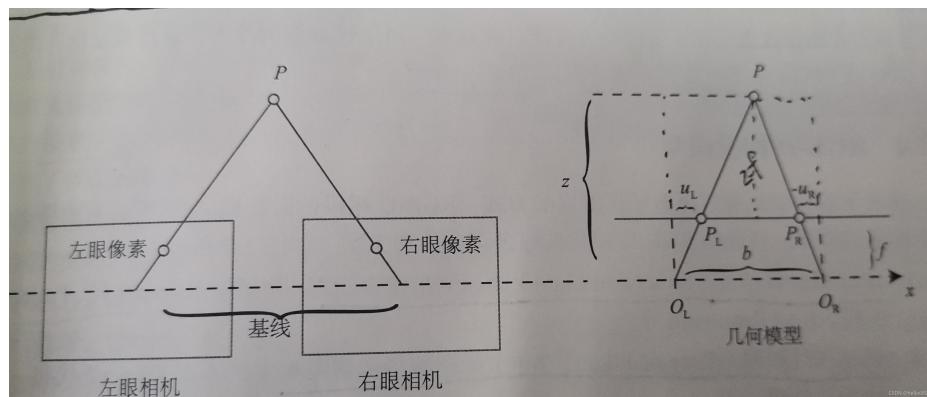


图 4.1 双目相机成像模型

由  $\Delta PP_LP_R$  与  $\Delta PO_LO_R$  相似得

$$\frac{z-f}{z} = \frac{b-u_L+u_R}{b} \quad (4.1)$$

其中焦距  $f$ ,  $b$  基线是相机参数, 已知。整理得:

$$z = \frac{fb}{d}, d = u_L - u_R \quad (4.2)$$

有针孔相机模型:

$$\begin{cases} u = f_x \frac{X}{Z} + c_x \\ v = f_y \frac{Y}{Z} + c_y \end{cases} \quad (4.3)$$

所以当由  $u, v, d \rightarrow X, Y, Z$  时，有：

$$\begin{cases} X = \frac{u - c_x}{f_x} * Z \\ Y = \frac{v - c_y}{f_y} * Z \\ Z = depth = \frac{fb}{d} \end{cases} \quad (4.4)$$

当由  $X, Y, Z \rightarrow u, v, d$  时，易得

$$\begin{cases} u = f_x \frac{X}{Z} + c_x \\ v = f_y \frac{Y}{Z} + c_y \\ d = \frac{fb}{Z} \end{cases} \quad (4.5)$$

不过一般双目相机没有深度  $Z$ ，所以需要借助双目深度恢复算法，如 SGBM 算法：

$$d = SGBM(X_{left}, Y_{left}, X_{Right}, Y_{Right}) \quad (4.6)$$

## 4.2 推导在右目相机下该模型将发生什么改变

使用右眼相机坐标减去左眼相机坐标， $d = X_r - X_l$ 。使用右眼相机的投影点，另外使用左眼相机的外参。(这部分不太懂)

实践部分：

## 4.3 根据双目图像和深度图绘制点云

核心代码如下：

```

1 // start your code here (~6 lines)
2 // 根据双目模型计算 point 的位置
3     double x = (u - cx) / fx; // 和单目一样，只是这里可以计算深度信息，所以可以构建点云
4     double y = (v - cy) / fy;
5     double depth = fx * b / (disparity.at<uchar>(v, u)); // z=fb/d
6     point[0] = x * depth;
7     point[1] = y * depth;
8     point[2] = depth;
9     pointcloud.push_back(point);
10    // end your code here

```

Listing 4: 工程/stereoVision.cpp

点云图如图 4.2 所示



图 4.2 双目生成点云图

## 5 第 5 题

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

对于  $\frac{\partial(\mathbf{Ax})}{\partial x}$ , 记  $\mathbf{f} = \mathbf{Ax}$ , 则对于向量微分:

$$d\mathbf{f} = \left( \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)^T d\mathbf{x} \quad (5.1)$$

有:

$$\begin{aligned} \text{vec}(d\mathbf{f}) &= d\mathbf{f} \\ &= \text{vec}(\mathbf{Adx}) \\ &= \text{vec}(\mathbf{Adx} \mathbf{I}_{1 \times 1}) \\ &= (\mathbf{I}_{1 \times 1} \otimes \mathbf{A}) \text{vec}(d\mathbf{x}) \\ &= \mathbf{Adx} \end{aligned} \quad (5.2)$$

对比 (5.2) 和 (5.3) 可得:

$$\frac{d(\mathbf{Ax})}{d\mathbf{x}} = \frac{\partial(\mathbf{Ax})}{\partial \mathbf{x}} = \mathbf{A}^T \quad (5.3)$$

## 5.2 求 $d(\mathbf{x}^T \mathbf{A} \mathbf{x})/d\mathbf{x}$

由题目条件可知  $f = \mathbf{x}^T \mathbf{A} \mathbf{x}$  是标量函数,

$$\begin{aligned}\frac{d(\mathbf{x}^T \mathbf{A} \mathbf{x})}{d\mathbf{x}} &= \frac{\partial \mathbf{x}^T \mathbf{A} \mathbf{x}}{\partial \mathbf{x}} \\ &= \frac{(\partial \mathbf{x})^T \mathbf{A} \mathbf{x} + \mathbf{x}^T \mathbf{A} \partial \mathbf{x}}{d\mathbf{x}} \\ &= \frac{\mathbf{x}^T \mathbf{A}^T d\mathbf{x} + \mathbf{x}^T \mathbf{A}^T d\mathbf{x}}{d\mathbf{x}} \\ &= \mathbf{x}^T (\mathbf{A} + \mathbf{A}^T)\end{aligned}\tag{5.4}$$

对于矩阵微分和迹有以下性质:

1. 标量套上述:  $a = \text{tr}(a)$
2. 转置:  $\text{tr}(A^T) = \text{tr}(A)$ 。
3. 线性:  $\text{tr}(A \pm B) = \text{tr}(A) \pm \text{tr}(B)$ 。
4. 矩阵乘法交换:  $\text{tr}(AB) = \text{tr}(BA)$ , 其中  $A$  与  $B^T$  尺寸相同。两侧都等于  $\sum_{i,j} A_{ij} B_{ji}$ 。
5. 矩阵乘法/逐元素乘法交换:  $\text{tr}(A^T(B \odot C)) = \text{tr}((A \odot B)^T C)$ , 其中  $A, B, C$  尺寸相同。两侧都等于  $\sum_{i,j} A_{ij} B_{ij} C_{ij}$ 。

$$(d\mathbf{X})^T = d\mathbf{X}^T\tag{5.5}$$

当  $\mathbf{A}$  为常数矩阵时:

$$d(\mathbf{X} \mathbf{A} \mathbf{X})^T = (d\mathbf{X})^T \mathbf{A} \mathbf{X}^T + \mathbf{X} \mathbf{A} (d\mathbf{X})^T\tag{5.6}$$

$$d(\mathbf{X}^T \mathbf{A} \mathbf{X}) = (d\mathbf{X})^T \mathbf{A} \mathbf{X} + \mathbf{X}^T \mathbf{A} d\mathbf{X}\tag{5.7}$$

对于  $f = \mathbf{x}^T \mathbf{A} \mathbf{x}$  有:

$$\begin{aligned}df &= \text{tr}(df) = \text{tr}((d\mathbf{x})^T \mathbf{A} \mathbf{x} + \mathbf{x}^T \mathbf{A} d\mathbf{x}) \\ &= \text{tr}((d\mathbf{x})^T \mathbf{A} \mathbf{x}) + \text{tr}(\mathbf{x}^T \mathbf{A} d\mathbf{x}) \\ &\stackrel{\text{tr}(AB)=\text{tr}(BA)}{=} \text{tr}((\mathbf{A} \mathbf{x})^T d\mathbf{x}) + \text{tr}(\mathbf{x}^T \mathbf{A} d\mathbf{x}) \\ &= \text{tr}(((\mathbf{A} \mathbf{x})^T + \mathbf{x}^T \mathbf{A}) d\mathbf{x})\end{aligned}\tag{5.8}$$

又由实值函数矩阵导数与微分的联系:

$$df = \text{tr}(df) = \text{tr}\left(\left(\frac{\partial f}{\partial \mathbf{x}}\right)^T d\mathbf{x}\right) \quad (5.9)$$

对比 (5.8) 和 (5.9) 得

$$\begin{aligned} \frac{d(\mathbf{x}^T \mathbf{A} \mathbf{x})}{d\mathbf{x}} &= \frac{\partial(\mathbf{x}^T \mathbf{A} \mathbf{x})}{\partial \mathbf{x}} \\ &= \frac{\partial f}{\partial \mathbf{x}} \\ &= ((\mathbf{A} \mathbf{x})^T + \mathbf{x}^T \mathbf{A})^T \\ &= \mathbf{A} \mathbf{x} + \mathbf{A}^T \mathbf{x} \\ &= (\mathbf{A} + \mathbf{A}^T) \mathbf{x} \end{aligned} \quad (5.10)$$

### 5.3 证明 $\mathbf{x}^T \mathbf{A} \mathbf{x} = \text{tr}(\mathbf{A} \mathbf{x} \mathbf{x}^T)$

证明如下:

$$\begin{aligned} \text{left} &= \mathbf{x}^T \mathbf{A} \mathbf{x} \\ &= \begin{bmatrix} x_1 & x_2 \cdots x_n \end{bmatrix} \cdot \begin{bmatrix} a_{11} & a_{12} \cdots a_{1n} \\ a_{21} & a_{22} \cdots a_{2n} \\ \vdots & \ddots \\ a_{n1} & a_{n2} \cdots a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \\ &= x_1 \sum_{i=1}^n a_{1i} x_i + x_2 \sum_{i=1}^n a_{2i} x_i + \cdots + x_n \sum_{i=1}^n a_{ni} x_i \end{aligned} \quad (5.11)$$

$$\begin{aligned}
tr(\mathbf{A} \mathbf{x} \mathbf{x}^T) &= tr\left(\begin{bmatrix} a_{11} & a_{12} \cdots a_{1n} \\ a_{21} & a_{22} \cdots a_{2n} \\ \vdots & \ddots \\ a_{n1} & a_{n2} \cdots a_{nn} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \cdot \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}\right) \\
&= tr\left(\begin{bmatrix} a_{11} & a_{12} \cdots a_{1n} \\ a_{21} & a_{22} \cdots a_{2n} \\ \vdots & \ddots \\ a_{n1} & a_{n2} \cdots a_{nn} \end{bmatrix} \begin{bmatrix} x_1^2 & x_1 x_2 & \cdots & x_1 x_n \\ x_2 x_1 & x_2^2 & \cdots & x_2 x_n \\ \vdots & \vdots & \ddots & \vdots \\ x_n x_1 & x_n x_2 & \cdots & x_n^2 \end{bmatrix}\right) \\
&= x_1 \sum_{i=1}^n a_{1i} x_i + x_2 \sum_{i=1}^n a_{2i} x_i + \cdots + x_n \sum_{i=1}^n a_{ni} x_i \\
&= left \tag{5.12}
\end{aligned}$$

得证。

## 6 第 6 题

定义误差为:

$$e_i = y_i - \exp(ax_i^2 + bx_i + c) \tag{6.1}$$

误差对每个状态变量求偏导，以构造雅可比：

$$\begin{cases} \frac{\partial e_i}{\partial a} = -x_i^2 \exp(ax_i^2 + bx_i + c) \\ \frac{\partial e_i}{\partial b} = -x_i \exp(ax_i^2 + bx_i + c) \\ \frac{\partial e_i}{\partial c} = -\exp(ax_i^2 + bx_i + c) \end{cases} \tag{6.2}$$

核心代码如下所示：

```

1   for (int i = 0; i < N; i++) {
2       double xi = x_data[i], yi = y_data[i]; // 第 i 个数据点
3       // start your code here
4       double error = yi - exp(ae * xi * xi + be * xi + ce); // 第 i 个
      数据点的计算误差
5       Vector3d J; // 雅可比矩阵(是一个列向量)
6       J[0] = -xi * xi * exp(ae * xi * xi + be * xi + ce); // de/da
7       J[1] = -xi * exp(ae * xi * xi + be * xi + ce); // de/db
8       J[2] = -exp(ae * xi * xi + be * xi + ce); // de/dc
9       // H x=b

```

```
10         H += J * J.transpose(); // GN近似的H
11         b += -error * J;
12         // end your code here
13
14         cost += error * error;
15     }
16
17     // 求解线性方程 Hx=b, 建议用ldlt
18     // start your code here
19     Vector3d dx = H.ldlt().solve(b); //MatrixBase的Cholesky分解求解线性
20     // end your code here
```

Listing 5: 工程/gaussnewton.cpp

```
1 cmake_minimum_required(VERSION 3.21)
2 project(T6)
3
4 set(CMAKE_CXX_STANDARD 11)
5
6 # OpenCV
7 find_package(OpenCV REQUIRED)
8 include_directories(${OpenCV_INCLUDE_DIRS})
9
10 # Eigen
11 include_directories("/usr/include/eigen3")
12
13 add_executable(gaussnewton gaussnewton.cpp)
14 target_link_libraries(gaussnewton ${OpenCV_LIBS})
```

Listing 6: 工程/CMakeLists.txt

```

/home/wrk/SLAM/DeepBlueCurriculum/VSLAM/ch4_camera_optimize/Francisrk-第4章作业/Pr
total cost: 3.19575e+06
total cost: 376785
total cost: 35673.6
total cost: 2195.01
total cost: 174.853
total cost: 102.78
total cost: 101.937
total cost: 101.937
total cost: 101.937
cost: 101.937, last cost: 101.937
estimated abc = 0.890912, 2.1719, 0.943629

Process finished with exit code 0
|
```

图 6.1 手写高斯牛顿法结果

## 7 第 7 题

### 7.1 给出 H 的具体形式

此处的运动和观测方程和 14 讲 P125 中的方程不一样，此处有

$$\begin{cases} x_k = x_{k-1} + v_k + w_k, & \omega \sim \mathcal{N}(0, Q) \\ y_k = x_k + n_k, & n_k \sim \mathcal{N}(0, R) \end{cases} \quad (7.1)$$

其中

$$\begin{cases} e_{v,k} = v_k - (x_k - x_{k-1}) \\ e_{y,k} = y_k - x_k \end{cases} \quad (7.2)$$

(14 讲中的  $e_{u,k}$  定义反了, 导致最后的  $H$  是错的。) 所以

$$\begin{aligned}
 \mathbf{e}_k &= \begin{bmatrix} e_{v,k} \\ e_{y,k} \end{bmatrix} \\
 &= \begin{bmatrix} v_k - (x_k - x_{k-1}) \\ y_k - x_k \end{bmatrix} \\
 &= \begin{bmatrix} v_k \\ y_k \end{bmatrix} - \begin{bmatrix} x_k - x_{k-1} \\ x_k \end{bmatrix} \\
 &= \mathbf{z}_k - \begin{bmatrix} x_k - x_{k-1} \\ x_k \end{bmatrix}
 \end{aligned} \tag{7.3}$$

所以

$$\mathbf{e} = \begin{bmatrix} \mathbf{e}_v \\ \mathbf{e}_y \end{bmatrix} = \begin{bmatrix} v_1 \\ v_2 \\ v_3 \\ y_1 \\ y_2 \\ y_3 \end{bmatrix} - \begin{bmatrix} x_1 - x_0 \\ x_2 - x_1 \\ x_3 - x_2 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \mathbf{z} - \mathbf{H}\mathbf{x} \tag{7.4}$$

由 (7.4) 得,  $\mathbf{H} \in \mathbb{R}^{6 \times 4}$  所以

$$\mathbf{H}\mathbf{x} = \mathbf{H} \cdot \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} x_1 - x_0 \\ x_2 - x_1 \\ x_3 - x_2 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \tag{7.5}$$

结合  $H$  的维度可得

$$\mathbf{H} = \begin{bmatrix} -1 & 1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

## 7.2 求 $W$ 的具体取值

由第 1 问可得：转化为最小二乘问题：

$$\begin{aligned}
 x^* &= \operatorname{argmin} \frac{1}{2} \mathbf{e}^T W^{-1} \mathbf{e} \\
 &= \operatorname{argmin} \left( \frac{1}{2} \left( \sum_{k=1}^3 e_{v,k}^T Q_k^{-1} e_{v,k} + \left( \sum_{k=1}^3 e_{y,k}^T R_k^{-1} e_{y,k} \right) \right) \right. \\
 &= \operatorname{argmin} \left( \frac{1}{2} \begin{bmatrix} e_{v,1}^T & e_{v,2}^T & e_{v,3}^T \end{bmatrix} \begin{bmatrix} Q_1^{-1} & & \\ & Q_2^{-1} & \\ & & Q_3^{-1} \end{bmatrix} \begin{bmatrix} e_{v,1} \\ e_{v,2} \\ e_{v,3} \end{bmatrix} + \right. \\
 &\quad \left. \begin{bmatrix} e_{y,1}^T & e_{y,2}^T & e_{y,3}^T \end{bmatrix} \begin{bmatrix} R_1^{-1} & & \\ & R_2^{-1} & \\ & & R_3^{-1} \end{bmatrix} \begin{bmatrix} e_{y,1} \\ e_{y,2} \\ e_{y,3} \end{bmatrix} \right) \\
 &= \operatorname{argmin} \left( \frac{1}{2} \begin{bmatrix} e_{v,1} \\ e_{v,2} \\ e_{v,3} \\ e_{y,1} \\ e_{y,2} \\ e_{y,3} \end{bmatrix}^T \begin{bmatrix} Q_1 & & & & & \\ & Q_2 & & & & \\ & & Q_3 & & & \\ & & & R_1 & & \\ & & & & R_2 & \\ & & & & & R_3 \end{bmatrix}^{-1} \begin{bmatrix} e_{v,1} \\ e_{v,2} \\ e_{v,3} \\ e_{y,1} \\ e_{y,2} \\ e_{y,3} \end{bmatrix} \right) \\
 &= \operatorname{argmin} \left( \frac{1}{2} \mathbf{e}^T \mathbf{W}^{-1} \mathbf{e} \right)
 \end{aligned} \tag{7.6}$$

所以

$$\mathbf{W} = \operatorname{diag}(Q_1, Q_2, Q_3, R_1, R_2, R_3) \tag{7.7}$$

## 7.3 是否有唯一解？

是否有唯一解就看梯度下降时其系数矩阵的导数 =0 是否有唯一解，

$$\text{记 } F = \frac{1}{2}(z - Hx)^T * W^{-1} * (z - Hx)$$

$$\begin{aligned}\frac{\partial F}{\partial x} &= \frac{1}{2} \frac{\partial(z^T W^{-1} z - z^T W^{-1} Hx - x^T H^T W^{-1} z + x^T H^T W^{-1} Hx)}{\partial x} \\ &= H^T W^{-1} Hx - H^T W^{-1} z\end{aligned}$$

则由  $H^T W^{-1} Hx = H^T W^{-1} z$ , 即  $Ax = b$  的形式, 接着证明  $A$  可逆, 即可证有唯一解

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

## 参考文献

- [1] [https://blog.csdn.net/qq\\_16137569/article/details/112398976#t5](https://blog.csdn.net/qq_16137569/article/details/112398976#t5)